

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.»

Филиал федерального государственного бюджетного образовательного
учреждения высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.» в г. Петровске

УТВЕРЖДАЮ
Директор филиала СГТУ
имени Гагарина Ю.А. в г. Петровске

«06»  20  г.



МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

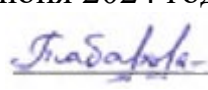
по дисциплине

ОП.04 «Основы алгоритмизации и программирования»

направление подготовки

09.02.07 «Информационные системы и программирование»

Методические указания рассмотрены
на заседании предметной (цикловой)
комиссии общепрофессиональных дисциплин,
профессиональных модулей специальностей
технического профиля
«14» июня 2024 года, протокол №12

Председатель ПЦК  /Ю.А.Табарова/

Петровск 2024

Пояснительная записка

Методические указания по выполнению практических работ подготовлены на основе рабочей программы учебной дисциплины ОП.04 «Основы алгоритмизации и программирования», разработанной на основе ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» и соответствующих общих (ОК) и профессиональных (ПК) компетенций:

ОК 01 - Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам,

ОК 02 -Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 04 - Эффективно взаимодействовать и работать в коллективе и команде.

ОК 05 - Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 09 - Пользоваться профессиональной документацией на государственном и иностранном языках.

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием.

ПК.1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять рефакторинг и оптимизацию программного кода.

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения.

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

При выполнении практических работ студент должен *знать*:

- понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции;
- эволюцию языков программирования, их классификацию, понятие системы программирования;
- основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти;
- подпрограммы, составление библиотек подпрограмм;

объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения

При выполнении практических работ студент должен *уметь*:

- разрабатывать алгоритмы для конкретных задач;

- использовать программы для графического отображения алгоритмов;
- определять сложность работы алгоритмов;
- работать в среде программирования;
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования;
- оформлять код программы в соответствии со стандартом кодирования;
- выполнять проверку, отладку кода программы.

Содержание практических занятий определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины.

Объем практических занятий по дисциплине определяется учебным планом по данной специальности.

Продолжительность практического занятия – 2 академических часа. Перед проведением практического занятия преподавателем организуется инструктаж, а по ее окончании – обсуждение итогов.

Комплект методических указаний по выполнению практических работ по дисциплине ОП.04 «Основы алгоритмизации и программирования» содержит 34 практических занятий.

Перечень практических работ

по дисциплине ОП.04 «Основы алгоритмизации и программирования»

ПРАКТИЧЕСКАЯ РАБОТА № 1.

Тема: Знакомство со средой программирования

ПРАКТИЧЕСКАЯ РАБОТА № 2.

Тема: Составление программ линейной структуры

ПРАКТИЧЕСКАЯ РАБОТА № 3.

Тема: Составление программ разветвляющей структуры

ПРАКТИЧЕСКАЯ РАБОТА № 4.

Тема: Составление программ циклической структуры

ПРАКТИЧЕСКАЯ РАБОТА № 5.

Тема: Обработка одномерных массивов

ПРАКТИЧЕСКАЯ РАБОТА № 6.

Тема: Обработка двумерных массивов

ПРАКТИЧЕСКАЯ РАБОТА № 7.

Тема: Работа со строками

ПРАКТИЧЕСКАЯ РАБОТА № 8.

Тема: Работа с данными типа множество

ПРАКТИЧЕСКАЯ РАБОТА № 9.

Тема: Составление программ с использованием текстовых файлов

ПРАКТИЧЕСКАЯ РАБОТА № 10.

Тема: Создание программ с использованием типизированных и нетипизированных файлов

ПРАКТИЧЕСКАЯ РАБОТА № 11.

Тема: Организация процедур и функций

ПРАКТИЧЕСКАЯ РАБОТА № 12.

Тема: Применение рекурсивных функций

ПРАКТИЧЕСКАЯ РАБОТА № 13.

Тема: Программирование модуля

ПРАКТИЧЕСКАЯ РАБОТА № 14.

Тема: Использование указателей для организации связанных списков

ПРАКТИЧЕСКАЯ РАБОТА № 15.

Тема: Создание проекта с использованием компонентов для работы с текстом

ПРАКТИЧЕСКАЯ РАБОТА № 16.

Тема: Создание проекта с использованием компонентов для работы с текстом

ПРАКТИЧЕСКАЯ РАБОТА № 17.

Тема: Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени

ПРАКТИЧЕСКАЯ РАБОТА № 18.

Тема: Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени

ПРАКТИЧЕСКАЯ РАБОТА № 19.

Тема: Создание проекта с использованием полос прокрутки для ввода информации.

ПРАКТИЧЕСКАЯ РАБОТА № 20.

Тема: Создание проекта с использованием группы зависимых переключателей

ПРАКТИЧЕСКАЯ РАБОТА № 21.

Тема: Создание процедур на основе событий

ПРАКТИЧЕСКАЯ РАБОТА № 22.

Тема: Создание процедур на основе событий

ПРАКТИЧЕСКАЯ РАБОТА № 23.

Тема: Создание проекта с использованием кнопочных компонентов

ПРАКТИЧЕСКАЯ РАБОТА № 24.

Тема: Создание проекта с использованием кнопочных компонентов

ПРАКТИЧЕСКАЯ РАБОТА № 25.

Тема: Создание проекта с использованием компонентов стандартных диалогов и системы меню.

ПРАКТИЧЕСКАЯ РАБОТА № 26.

Тема: Создание проекта с использованием компонентов стандартных диалогов и системы меню.

ПРАКТИЧЕСКАЯ РАБОТА № 27.

Тема: Разработка функциональной схемы работы приложения

ПРАКТИЧЕСКАЯ РАБОТА № 28.

Тема: Разработка функциональной схемы работы приложения

ПРАКТИЧЕСКАЯ РАБОТА № 29.

Тема: Разработка оконного приложения с несколькими формами

ПРАКТИЧЕСКАЯ РАБОТА № 30.

Тема: Разработка оконного приложения с несколькими формами

ПРАКТИЧЕСКАЯ РАБОТА № 31.

Тема: Разработка игрового приложения

ПРАКТИЧЕСКАЯ РАБОТА № 32.

Тема: Разработка игрового приложения

ПРАКТИЧЕСКАЯ РАБОТА № 33.

Тема: Разработка многооконного приложения. Компиляция и запуск приложения

ПРАКТИЧЕСКАЯ РАБОТА № 34.

Тема: Разработка многооконного приложения. Компиляция и запуск приложения

ИНСТРУКЦИИ ДЛЯ ОБУЧАЮЩИХСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

Прежде чем приступить к выполнению заданий, внимательно прочитайте данные рекомендации.

В ходе выполнения практических работ студент должен:

- выполнять требования по охране труда
- соблюдать инструкцию по правилам и мерам безопасности в кабинете информационных технологий
- строго выполнять весь объем работы, указанный в задании
- соблюдать требования эксплуатации компьютерной техники (правила включения и выключения)
- предоставить отчет о проделанной работе по окончании выполненной работы, который должен содержать:

1. Название работы.
2. Цель работы.
3. Задание и его решение.
4. Вывод о проделанной работе.

Текст отчета по практической работе должен быть набран на компьютере шрифтом Times New Roman размером 14 пт. (при оформлении текста используется текстовый редактор MS Word). Шрифт, используемый в иллюстративном материале (таблицы и рисунки), рекомендуется уменьшить до 12 пт. Межстрочный интервал в основном тексте - полуторный. В иллюстративном материале межстрочный интервал рекомендуется сделать одинарным. Поля страницы должны быть: левое поле - 30 мм; правое поле – 15 мм; верхнее и нижнее поле - 20 мм.

Каждый абзац должен начинаться с красной строки. Отступ абзаца – 1,25 см от левой границы текста.

Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием). Практическая работа выполняется согласно заданию и методическим рекомендациям. Результат работы представляется преподавателю в виде файла (файлов) в личном каталоге, защищается обучающимися.

По ходу выполнения работы при возникновении вопросов обучающийся может получить консультацию у преподавателя или самостоятельно воспользоваться лекционным материалом, рекомендуемой литературой.

1. Составление программы на языке программирования

Правила оформления кода:

1. Используйте разумные имена для переменных и функций

Программа должна быть хорошо понятна человеку при чтении. Если при чтении программы приходится понимать назначение переменных и функций по тому, как они используются, то читать код становится гораздо сложнее. Неудачно выбранные имена могут привести к тому, что смысл программы может быть неправильно понят. Выбирайте такие имена, которые бы объясняли

смысл переменных и функций, тогда код станет гораздо понятнее, и не потребуется писать множество комментариев.

При написании составных слов, например в именах переменных, пишите их слитно без пробелов, при этом каждое новое слово пишется с большой буквы.

2. Не дублируйте код. Если в программе есть одинаковые выражения или фрагмента кода, вынесите этот код в отдельную функцию. Верным признаком необходимости создания новой функции является желание скопировать фрагмент кода из одного места программы в другое. В таком случае сразу перенесите этот фрагмент в отдельную функцию.

Если фрагменты кода похожи, но не идентичны, подумайте, не получится ли и их вынести в одну функцию, возможно добавив параметры или условия.

3. Не используйте «магические константы». Использование неименованных «магических» констант в коде нежелательно:

- при чтении кода может быть не понятно, что это за число, и почему оно именно такое;
- чаще всего одно и то же число потребуется написать в нескольких местах кода. Если его придётся изменять, можно пропустить одно из использований, что приведёт к ошибке.

Если в коде нужно использовать константу, дайте ей имя, используя `const`.

4. Расставляйте пробелы вокруг бинарных операторов. Это улучшает читаемость формул.

5. Всегда выделяйте блоки условных операторов и циклов скобками. В любой блок условия или цикла может захотеться добавить новое выражение. При этом можно забыть добавить скобки. Лучше сразу добавить скобки, чтобы потом не было с этим проблем.

6. Расставляйте скобки одинаково. Выберите и используйте для себя один из стилей расстановки скобок. Это улучшает читаемость структуры программы.

7. Не делайте строки слишком длинными. Строка программы должна помещаться на экране. Обычно рекомендуют ограничить максимальную ширину строки в 80 символов. Если определение или вызов функции получается слишком широким поместите по одному параметру на каждой строке. Длинные математические выражения разбивайте на несколько строк, разбивая по границам логических блоков выражения.

8. Объявляйте переменные непосредственно перед использованием. Обязательно указывайте начальное значение для переменных. Значение неинициализированной переменной может быть любым. Использование (чтение) такого значения приведёт к недетерминированной работе программы, а в некоторых случаях является неопределённым поведением. Чтобы избежать проблем всегда инициализируйте переменные прямо в момент их создания.

9. Единый стиль оформления кода во всем проекте.

10. Визуальное выделение наиболее значимых частей — используя *вертикальное форматирование*, мы выделяем объявление переменных, цикл заполнения массива случайными числами и цикл обработки

по формуле. Если ваша функция выполняет несколько действий — то разумно разделить соответствующие блоки кода пустыми строками.

ПРАКТИЧЕСКАЯ РАБОТА №1

Тема: Знакомство со средой программирования

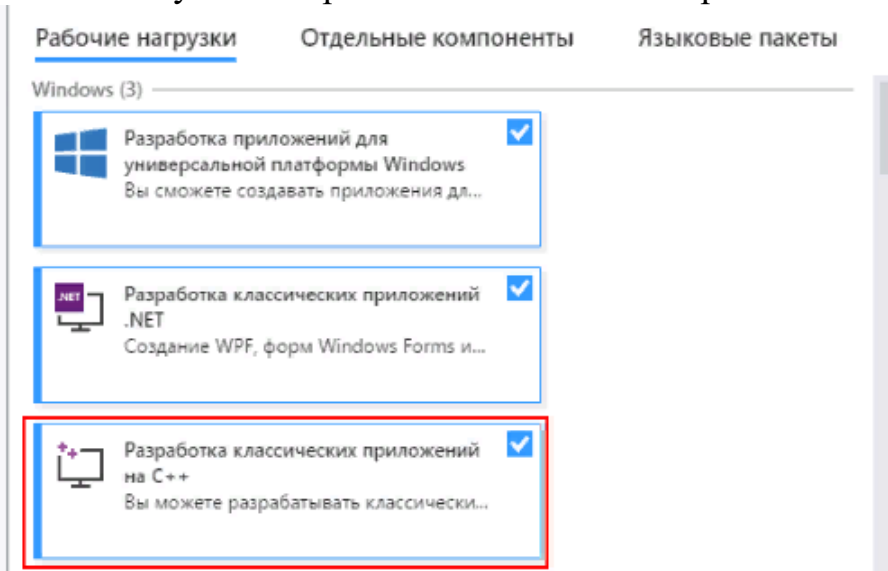
Цель работы: сформировать практические навыки работы с системой Visual Studio; научиться создавать, вводить в компьютер, выполнять и исправлять простейшие программы на языке C++

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

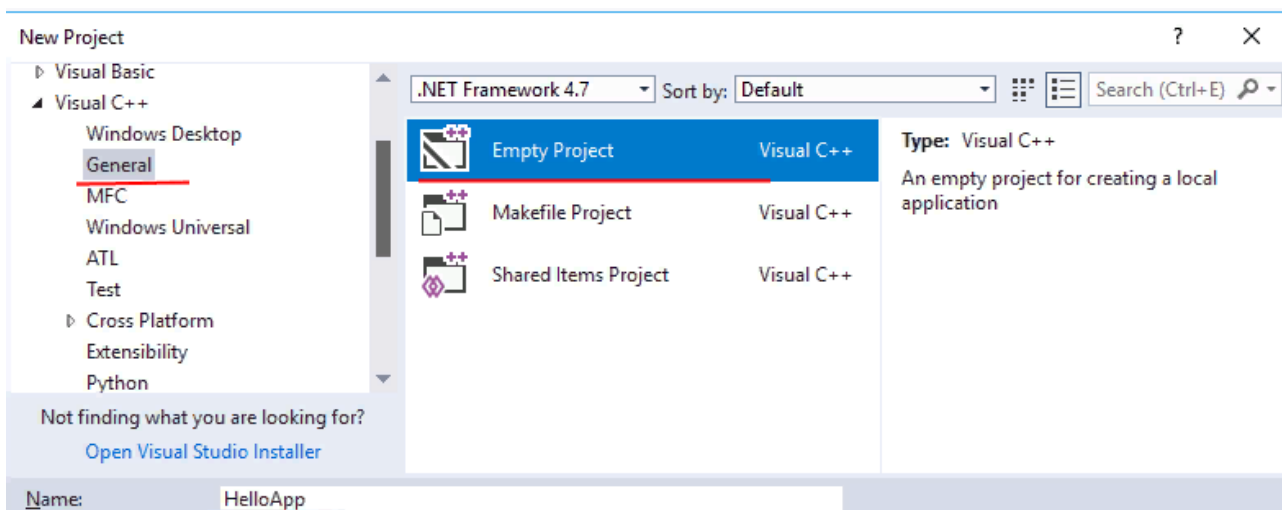
Содержание работы:

Задание 1. Познакомьтесь со способами загрузки среды программирования Visual Studio, её интерфейсом. Изучите назначение команд меню системы программирования Visual Studio.

1. После загрузки и запуска установщика Visual Studio в нем необходимо отметить пункт "Разработка классических приложений на C++":

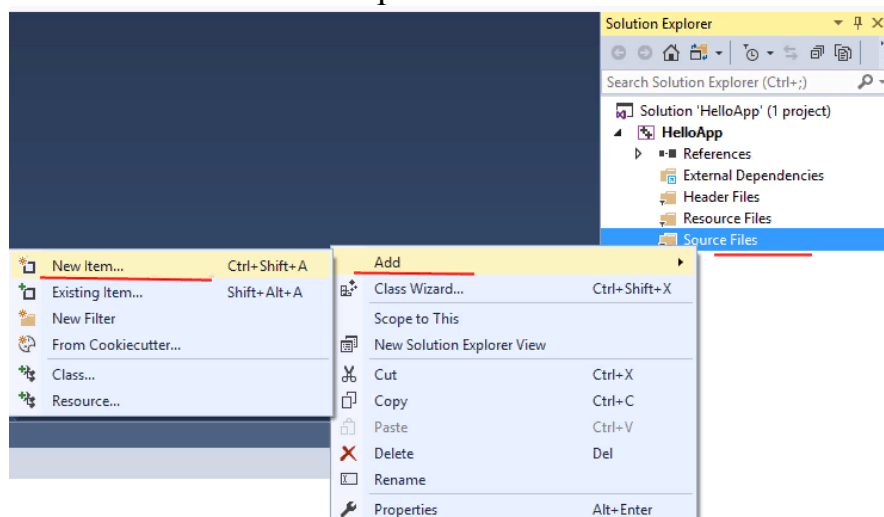


2. Выбрав все необходимые пункты, нажмем ОК для запуска установки. После установки Visual Studio создадим первый проект. Для этого перейдем в меню File (Файл) -> New (Создать) -> Project... (Проект), и нам откроется окно создания нового проекта. В нем перейдем в левой части окна к языку C++ и выберем его подсекцию General:

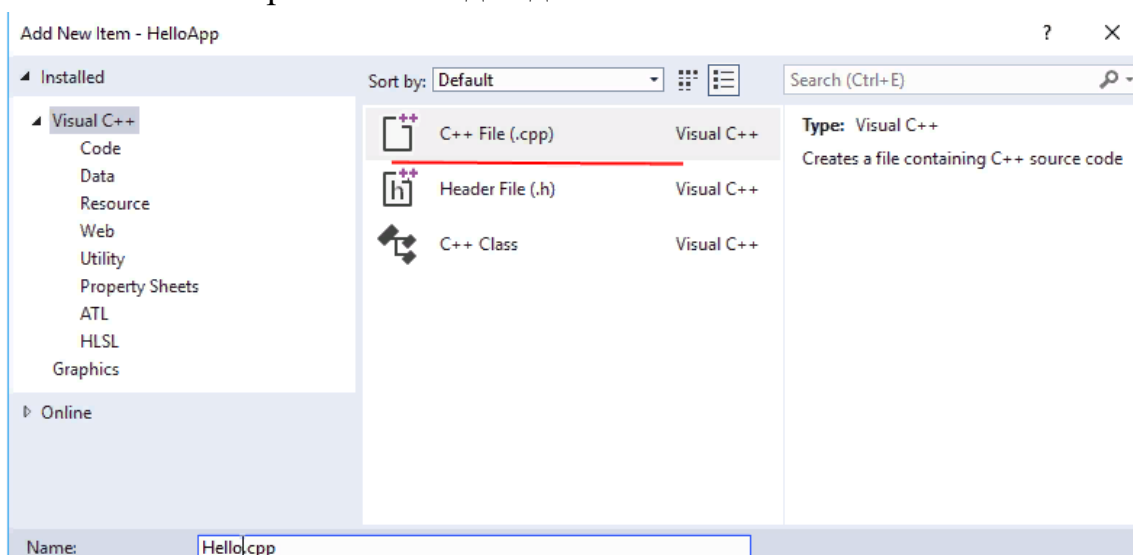


3. В центральной части окна в качестве типа проекта выберем Empty Project, а внизу окна в поле для имени проекта дадим проекту имя HelloApp и нажмем на ОК для создания проекта.

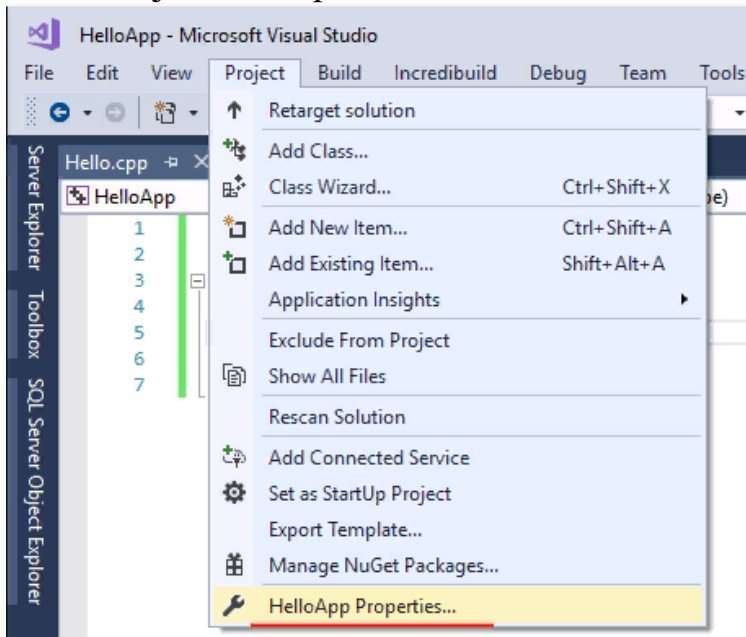
После этого Visual Studio создаст пустой проект. Добавим в него текстовый файл для набора исходного кода. Для этого в окне Solution Explorer (Обозреватель решений) нажмем правой кнопкой мыши на узел Source Files и в контекстном меню выберем Add -> New Item...:



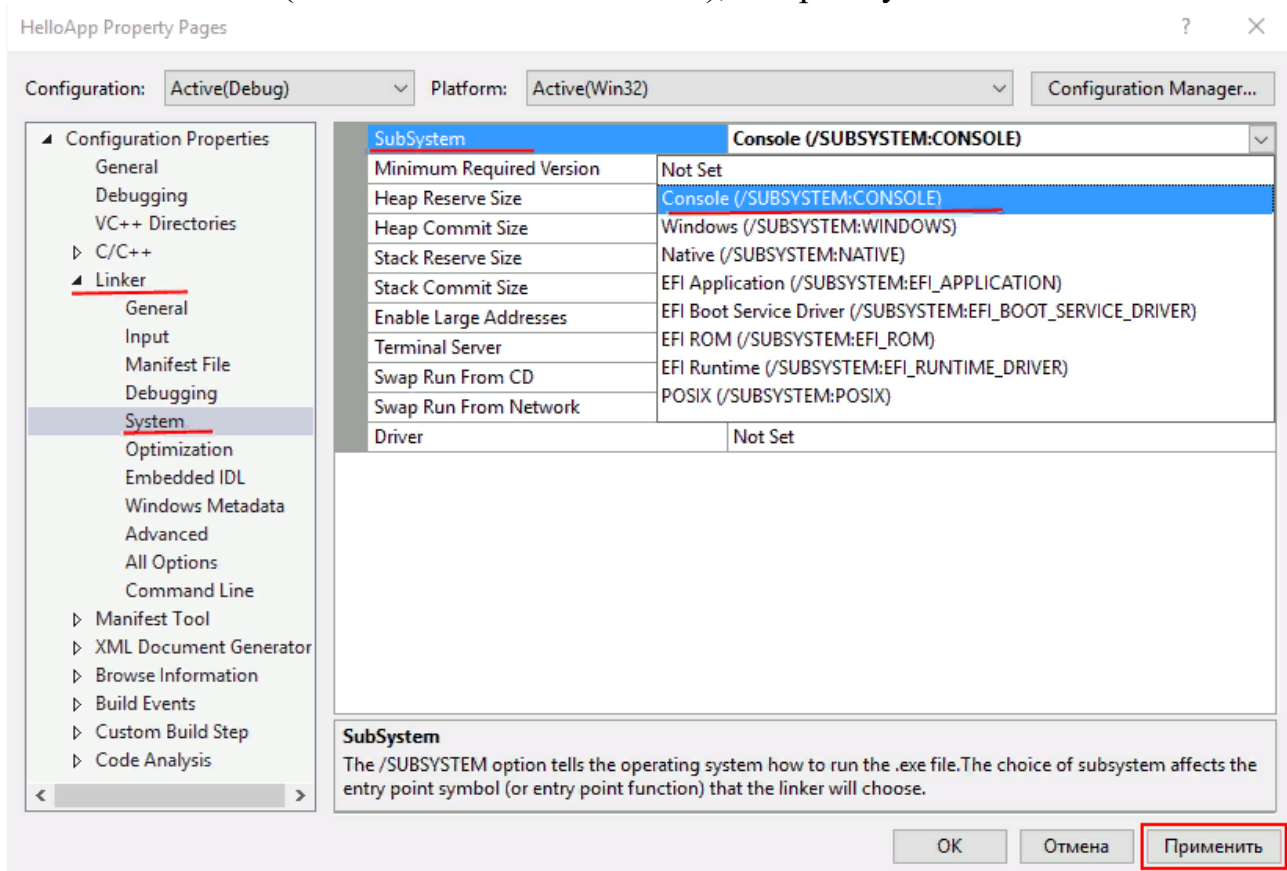
4. Затем нам откроется окно для добавления нового элемента:



5. Здесь нам надо выбрать пункт C++ File(.cpp), а внизу окна укажем для файла имя Hello.cpp. Как правило, исходные файлы на C++ имеют расширение .cpp. После добавления файла изменим опции проекта. Для этого перейдем к пункту меню Project -> Properties:

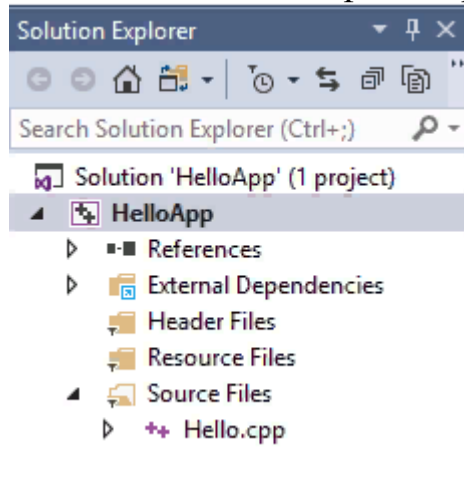


6. И в открывшемся окне свойств проекта в левой части перейдем к секции Linker -> System и далее для поля SubSystem установим значение Console(/SUBSYSTEM:CONSOLE), выбрав нужный элемент в списке:



Тем самым мы указываем, что мы хотим создать консольное приложение. После установки этого значения нажмем на кнопку "Применить", чтобы новые настройки конфигурации вступили в силу.

7. После добавления файла проект будет иметь следующую структуру:



Задание 2. Объявить переменные, с помощью которых можно будет посчитать общую сумму покупки нескольких товаров. Например, плитки шоколада, кофе и пакеты молока.

```
#include <iostream>
using namespace std;
int main() {
    setlocale(LC_ALL, "rus");
    int chocolate = 2; // хранит количество упаковок
    int milk = 3;
    int coffee = 1;
    float priceOfChocolate = 11.04; // хранит цены за одну упаковку
    float priceOfMilk = 9.59;
    float priceOfCoffee = 70.77;
    float sum = 0; // общая сумма покупки
    // считаем стоимость
    sum = (chocolate * priceOfChocolate) + (milk * priceOfMilk) + (coffee *
priceOfCoffee);
    cout << "Общая стоимость покупки = "; // показываем расчет и общую
стоимость на экран
    cout << chocolate * priceOfChocolate << '+' << milk * priceOfMilk << '+' <<
coffee * priceOfCoffee;
    cout << " = " << sum << endl << endl;
    return 0; }
```

Задание 3. Объявить три переменные типа `int` и присвоить первой числовое значение, вторая переменная равна первой переменной увеличенной на 3, а третья переменная равна сумме первых двух.

Задание 4: Объявить переменные, для подсчета общего количества предметов для сервировки стола. Например, чашки, такое же количество блюд и ложек.

ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема: Составление программ линейной структуры

Цель работы: Научиться составлять программы линейной структуры, реализовывать в программе оператор присваивания, процедуры ввода/вывода; строить блок-схемы линейной конструкции.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1: Создайте 4 переменные с разными типами данных и предложите пользователю ввести в них значения. После ввода, отобразите их на экране.

```
#include <iostream>
#include <locale>
using namespace std;
int main() {
    setlocale(LC_ALL, "rus");
    int digit = 0;
    double digit2 = 0;
    char symbol = 0;
    bool trueOrFalse = 0;
    cout << "Введите целое число: ";
    cin >> digit;
    cout << "Введите вещественное число: ";
    cin >> digit2;
    cout << "Введите символ: ";
    cin >> symbol;
    // в переменную типа bool с помощью cin можно ввести
    // только числа 0 (интерпретируется как false) и 1 (true)
    cout << "Введите 0 или 1: ";
    cin >> trueOrFalse;
    cout << endl << endl;
    cout << "Целое число: " << digit << endl;
    cout << "Вещественное число: " << digit2 << endl;
    cout << "Символ: " << symbol << endl;
    cout << "bool: " << trueOrFalse << endl;
    return 0;
}
```

Задание 2. Дано четырехзначное число (к примеру 5678), вывести на экран в обратном порядке цифры из которых это число состоит. То есть мы

должны увидеть на экране 8765. Подсказка: чтобы взять из числа отдельные цифры, надо применять деление по модулю на 10.

```
#include <iostream>
using namespace std;
int main(){
    setlocale(LC_ALL, "rus");
    int mainNumber = 5678;
    cout << "Дано целое число: " << mainNumber << endl;
    cout << "Число наизнанку: ";
    // остаток от деления четырехзначного числа 5678 на 10
    cout << mainNumber % 10; // 5678 % 10 = 8
    // далее делим mainNumber на 10 и записываем в переменную
    // так как тип переменной int, дробная часть отбросится
    // и mainNumber будет равен 567 (а не 567,8)
    mainNumber /= 10;
    // показываем остаток от деления 567 на 10 на экран
    cout << mainNumber % 10;
    mainNumber /= 10;
    cout << mainNumber % 10;
    mainNumber /= 10;
    cout << mainNumber % 10;
    mainNumber /= 10;
    cout << endl << endl;
    return 0;}
```

Задание 3. Написать программу пересчета расстояния из миль в километры (1 миля равна 1600,94 м)

Задание 4. Написать программу, которая выводит на экран фразу «Каждый охотник желает знать, где сидит фазан», каждое слово с новой строки

Задание 5. Составить программу для вычисления среднего значения n чисел

Задание 6. Мальчик купил несколько тетрадей по сто рублей и несколько обложек по 50 рублей. Составить программу, которая могла бы подсчитать стоимость всей покупки.

Задание 7. Написать программу вычисления объема куба

Задание 8. Написать программу вычисления стоимости некоторого количества (по весу), например яблок

Задание 9. Написать программу для суммирования n чисел.

Задание 10. Написать программу вычисления площади параллелограмма

ПРАКТИЧЕСКАЯ РАБОТА № 3

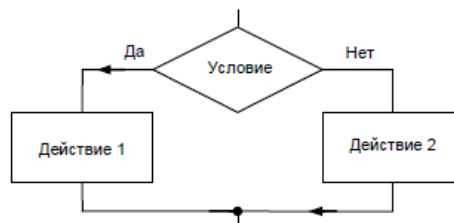
Тема: Составление программ разветвляющей структуры

Цель работы: Научиться реализовывать алгоритмическую конструкцию ветвление с помощью условного оператора и оператора выбора в программе, написанной на языке C++.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Справочный материал:

- Инструкция `if` используется для выбора одного из двух направлений дальнейшего хода программы.
- Алгоритм, реализуемый инструкцией `if`, выглядит так:



- Выбор действия осуществляется в зависимости от значения *условия* — заключенного в скобки выражения, записанного после `if`.
- Инструкция, записанная после `else`, выполняется в том случае, если значение выражения *условие* равно нулю, во всех остальных случаях выполняется инструкция, следующая за условием.
- Если при выполнении (или невыполнении) условия требуется выполнить несколько инструкций программы, то эти инструкции следует объединить в группу — заключить в фигурные скобки.

Содержание работы:

Задание 1. Написать программу вычисления дохода по вкладу. Исходные данные: сумма и срок вклада. Процентная ставка зависит от суммы. Если сумма меньше 5000 руб., то процентная ставка 10%, если больше, то 13%.

```
// Доход по вкладу
#include <stdafx.h>
#include <conio.h>
#include <clocale>
int main() {
    float sum; // сумма вклада
    int period; // срок
    float percent; // процент, зависит от суммы
    float profit; // доход
    float total; // сумма в конце срока вклада
    setlocale (LC_ALL, "Rus");
```

```

printf("\nДОХОД\n");
printf("Сумма, руб. -> ");
scanf("%f",&sum);
printf("Срок вклада, мес. -> ");
scanf("%i",&period);
if ( sum < 5000)
percent = 10;
else
percent = 13;
profit = sum * percent/100/12 * period;
total = sum + profit;
printf("\nСумма: %3.2f руб.", sum);
printf("\nСрок вклада: %i мес.", period);
printf("\nПроцент годовой: %2.2f", percent);
printf("\nДоход: %3.2f руб.", profit);
printf("\nСумма в конце срока вклада: %3.2f руб.", total);
printf("\n\nДля завершения нажмите <Enter>");
getch();
return 0; }

```

Задание 2. Составить расписание на неделю. Пользователь вводит порядковый номер дня недели и у него на экране отображается, то, что запланировано на этот день.

```

#include <iostream>
using namespace std;
int main(){
    setlocale(LC_ALL, "rus");
    int dayNumber; // будет хранить выбор пользователя
    cout << "Введите день недели (1, 2, 3...): ";
    cin >> dayNumber; // ввод значения
    switch (dayNumber)
    {
        case 1:
            cout << "Понедельник: \n8:00 Работа \n19:00 Тренировка \n";
            break;
        case 2:
            cout << "Вторник: \n8:00 Работа \n";
            break;
        case 3:
            cout << "Среда: \n8:00 Работа \n19:00 Бассейн \n";
            break;
        case 4:
            cout << "Четверг: \n8:00 Работа \n20:00 ДР Бабушки \n";
            break;
        case 5:
            cout << "Пятница: \n8:00 Работа \n17:00 Тренировка \n";

```

```

        break;
    case 6:
        cout << "Суббота: \nПикник \n";
        break;
    case 7:
        cout << "Воскресенье: \nЧто угодно \n";
        break;
    default:
        cout << "Нет такого дня недели ))" << endl;
    }
    return 0;}

```

Задание 3. Написать программу вычисления стоимости печати фотографий. Формат фотографий 9*12 или 10*15. Если количество фотографий больше 10, то заказчику предоставляется скидка 5%.

Задание 4. Написать программу решения квадратного уравнения. Программа должна проверять правильность исходных данных и в случае, если коэффициент при второй степени неизвестного равен нулю, выводить соответствующее сообщение.

Задание 5. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется, если сумма покупки больше 500 руб, в 5% — если сумма больше 1000 руб.

Задание 6. Пользователь вводит порядковый номер пальца руки. Необходимо показать его название на экран

Задание 7. Написать программу проверки знания даты основания Петровска. В случае неправильного ответа пользователя, программа должна выводить правильную дату.

Задание 8. Написать программу, которая проверяет, делится ли на 3 введенное с клавиатуры целое число.

Задание 9. Написать программу, которая запрашивает у пользователя номер дня недели и затем выводит его название. Если введены неправильные данные, программа должна вывести сообщение об ошибке.

Задание 10. Написать программу, которая позволяет рассчитать стоимость ремонта в комнате. Исходные данные: размер комнаты, вид материала (обои, краска, плитка и т.д.)

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: Составление программ циклической структуры

Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с циклической структурой.

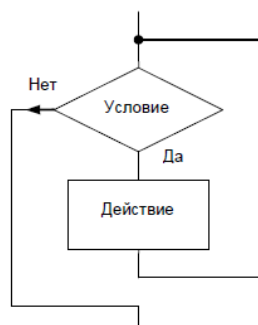
Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Справочный материал:

- Инструкция for используется для организации циклов с фиксированным числом повторений.
- Алгоритм, реализуемый инструкцией for, выглядит так:

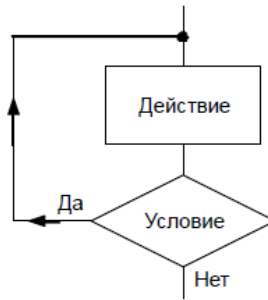


- Количество повторений цикла определяется начальным значением переменной-счетчика и условием завершения цикла.
- Переменная-счетчик должна быть целого типа и может быть объявлена непосредственно в инструкции цикла.
- While — это цикл с предусловием, т. е. возможна ситуация, при которой инструкции тела цикла не будут выполнены ни разу.
- Алгоритм, реализуемый инструкцией do while, выглядит так:



- Инструкции цикла while выполняются до тех пор, пока условие истинно (значение выражения Условие не равно нулю).
- Для того чтобы цикл while завершился, в его теле должны быть инструкции, влияющие на значения переменных, входящих в условие выполнения цикла.
- Цикл while обычно используется для организации приближенных вычислений, в задачах поиска и обработки данных, вводимых с клавиатуры или из файла.

- Цикл `do while` — это цикл с постусловием, т. е. инструкции тела цикла (между `do` и `while`) будут выполнены хотя бы один раз.
- Алгоритм, реализуемый инструкцией `do while`, выглядит так:



- После слова `while` записывается условие повторного выполнения инструкций цикла.
- Число повторений инструкций цикла `do while` определяется ходом выполнения программы.
- Для завершения цикла `do while` в теле цикла обязательно должны быть инструкции, выполнение которых влияет на условие завершения цикла.
- Цикл `do while`, как правило, используется для организации приближенных вычислений, в задачах поиска и обработки данных, вводимых с клавиатуры или из файла.

Содержание работы:

Задание 1. Написать программу, которая выводит таблицу значений функции $y = -2,4x^2 + 5x - 3$ в диапазоне от -2 до 2 , с шагом $0,5$.

// Таблица функции

```
#include "stdafx.h"
```

```
#include <conio.h>
```

```
#define LB -2.0 // нижняя граница диапазона изменения аргумента
```

```
#define HB 2.0 // верхняя граница диапазона изменения аргумента
```

```
#define DX 0.5 // приращение аргумента
```

```
int main() {
```

```
    float x,y; // аргумент и значение функции
```

```
    int n; // кол-во точек
```

```
    int i; // счетчик циклов
```

```
    n = (HB - LB)/DX + 1;
```

```
    x = LB;
```

```
    printf("-----\n");
```

```
    printf(" x | y\n");
```

```
    printf("-----\n");
```

```
    for (i = 1; i<=n; i++) {
```

```
        y = -2.4*x*x+5*x-3;
```

```
        printf("%6.2f | %6.2f\n",x,y);
```

```
        x += DX;    }
```

```
    printf("-----\n");
```

```
    printf("\nДля завершения нажмите <Enter>");
```

```
    getch();
```

```
return 0;    }
```

Задание 2. Необходимо суммировать все нечётные целые числа в диапазоне, который введёт пользователь с клавиатуры.

```
#include <iostream>
using namespace std;
int main(){
    setlocale(LC_ALL, "rus");
    int start = 0; // начало д-на
    int finish = 0; // конец д-на
    int sumUneven = 0;
    cout << "Введите начало диапазона: ";
    cin >> start;
    cout << "Введите конец диапазона: ";
    cin >> finish;
    int i = start; // управляющая переменная
    while (i <= finish) {
        if (i % 2 != 0) {
            cout << i << " "; // показать нечетные через пробел
            sumUneven += i; } // накапливать их сумму
        i++;}
    cout << "\nСумма нечетных чисел в диапазоне от " << start << " по "
<< finish;
    cout << " = " << sumUneven << endl << endl;
    return 0;}
```

Задание 3. Написать программу, вычисляющую сумму и среднее арифметическое последовательности положительных чисел, которые вводятся с клавиатуры.

```
#include <stdio.h>
#include <conio.h>
int main(){
    int a; // число, введенное с клавиатуры
    int n; // количество чисел
    int s; // сумма чисел
    float m; // среднее арифметическое
    s = 0;
    n = 0;
    printf("\Вычисление среднего арифметического");
    printf("последовательности положительных чисел.\n");
    printf("Вводите числа. Для завершения введите ноль.\n");
    do {
        printf("-> ");
        scanf("%i", &a);
        if (a > 0){
```

```

s += a;
n++;} }
while (a > 0);
printf("Введено чисел: %i\n", n);
printf("Сумма чисел: %i\n", s);
m = (float) s / n;
printf("Среднее арифметическое: %3.2f", m);
printf("\n\nДля завершения нажмите <Enter>");
getch();
return 0; }

```

Задание 4. На складе имеется определённое количество ящиков с яблоками (в нашем примере 15). Когда подъезжает машина для погрузки, попросить пользователя ввести, сколько ящиков загрузить в первую машину, во вторую и так далее, пока не закончатся ящики с яблоками. Предусмотреть тот случай, когда пользователь введёт количество ящиков больше, чем есть на складе.

Задание 5. Написать программу, которая выводит на экран таблицу стоимости, например, яблок в диапазоне от 100 г до 1 кг с шагом 100 г

Задание 6. Написать программу, которая выводит таблицу квадратов первых десяти целых положительных чисел

Задание 7. Написать программу, которая выводит на экран таблицу значения функции $y = 2^{x^2} - 5x - 8$ в диапазоне от -4 до 4 . Шаг изменения аргумента равен $0,5$

Задание 8. Из заданного натурального числа n удалить все четные цифры.

Задание 9. Написать программу, которая выводит на экран таблицу соответствия температуры в градусах Цельсия и Фаренгейта ($F^{\circ} = 5/9 \cdot C^{\circ} + 32$). Диапазон изменения температуры в градусах Цельсия и шаг должны вводиться во время работы программы

Задание 10. Написать программу, которая определяет минимальное число во введенной с клавиатуры последовательности положительных чисел (длина последовательности не ограничена).

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: Обработка одномерных массивов

Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с использованием одномерных массивов.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Написать программу, которая записывает введенные с клавиатуры данные в одномерный массив целого типа, состоящий из семи элементов. Перед вводом каждого элемента должна выводиться подсказка с номером элемента. После ввода последнего элемента программа должна вывести введенный массив и вычислить среднее арифметическое его элементов.

```
#include <stdio.h>
#include <conio.h>
int main() {
    int a[7]; // массив
    int sum; // сумма элементов массива
    float m; // среднее арифметическое
    int j;
    printf("\nВвод массива целых чисел");
    printf("После ввода каждого числа нажмите <Enter>\n");
    // ввод массива
    for (j = 0; j < 7; j++){
        printf("a[%i] -> ", j);
        scanf("%i",&a[j]); }
    // вывод массива
    printf("\nМассив: \n");
    for (j = 0; j < 7; j++) {
        printf("%i ", a[j]); }
    sum = 0;
    // вычислить сумму элементов
    for (j = 0; j < 7; j++) {
        sum = sum + a[j]; }
    m = sum / 7;
    printf("Среднее арифметическое: %f", m);
    printf("\nДля завершения нажмите <Enter>");
    getch();
    return 0;}
```

Задание 2. Создать массив типа int на 10 элементов и заполнить его случайными числами от 7 до 14. После заполнения перезаписать все числа,

которые больше десяти: от хранимого значение отнять 10. Например, в ячейке хранится число 12: $12 - 10 = 2$. Записать в эту ячейку 2.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
using namespace std;
int main(){
    int ourArr[10] = {};
    const int lowerLimit = 7;
    const int upperLimit = 14;
    srand(time(NULL));
    for(int i = 0; i < 10; i++) {        // заполняем и показываем
        ourArr[i] = lowerLimit + rand() % (upperLimit - lowerLimit + 1);
        cout << ourArr[i] << " | ";    }
    cout << endl << endl;
    for(int i = 0; i < 10; i++) {
        if(ourArr[i] >= 10)            {
            ourArr[i] -= 10;          }
        cout << ourArr[i] << " | ";    }
    cout << endl << endl;
    return 0;}
```

Задача 3. Заполнить массив из 50-ти элементов нечётными числами от 1 до 99. (используйте операцию остаток от деления, чтобы проверить число на чётность).

Задание 4. Написать программу, которая вводит с клавиатуры данные в одномерный массив дробного типа, состоящий из пяти элементов, после чего выводит количество ненулевых элементов. Перед вводом каждого элемента должна выводиться подсказка с номером элемента.

Задание 5. Написать программу, которая выводит минимальный элемент введенного с клавиатуры массива целых чисел.

Задание 6. Написать программу, которая вычисляет среднюю (за неделю) температуру воздуха. Исходные данные должны вводиться во время работы программы

Задание 7. Написать программу, которая проверяет, находится ли введенное с клавиатуры число в массиве

Задание 8. Написать программу, которая определяет количество учеников в классе, чей рост превышает средний.

Задание 9. Написать программу, суммирующую все элементы одномерного массива.

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: Обработка двумерных массивов

Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с использованием двумерных массивов.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Объявить двумерный массив и заполнить его построчно с клавиатуры. После заполнения – показать заполненную матрицу на экран и посчитать сумму элементов отдельно в каждом столбце и каждой строке.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
    setlocale(LC_ALL, "rus");
    const int MatrixSize = 3;
    int ourMatrix[MatrixSize][MatrixSize] = {};
    int rowSum[MatrixSize] = {}; // для записи суммы по строкам
    int columnSum[MatrixSize] = {}; // ... по столбцам
    cout << "Заполните матрицу " << MatrixSize << 'x' << MatrixSize << "
числами построчно.\n";
    for (int rowNum = 0; rowNum < MatrixSize; rowNum++) {
        for (int columnNum = 0; columnNum < MatrixSize; columnNum++) {
            cout << rowNum + 1 << "-я строка ";
            cout << columnNum + 1 << "-й столбец: ";
            cin >> ourMatrix[rowNum][columnNum];
        }
        cout << endl;
    }
    // вывод на экран + подсчет суммы
    for (int rowNum = 0; rowNum < MatrixSize; rowNum++) {
        cout << " | ";
        for (int columnNum = 0; columnNum < MatrixSize; columnNum++) {
            cout << setw(4) << ourMatrix[rowNum][columnNum] << " ";
            rowSum[rowNum] += ourMatrix[rowNum][columnNum];
            columnSum[columnNum] += ourMatrix[columnNum][rowNum];
        }
        cout << "|" << endl;
    }
    cout << "\nСумма по строкам!\n";
    for (int i = 0; i < MatrixSize; i++) {
        cout << i + 1 << "-я строка: " << rowSum[i] << endl;
    }
    cout << "\nСумма по столбцам!\n";
    for (int i = 0; i < MatrixSize; i++) {
        cout << i + 1 << "-й столбец: " << columnSum[i] << endl;
    }
    return 0;}
```

Задание 2. Объявить двумерный массив, заполнить целыми числами и показать на экран

Задание 3. Нарисовать таблицу умножения в консольном приложении

Задание 4. Заполнить двумерный массив случайными числами от 10 до 99. Посчитать сумму элементов отдельно в каждой строке и определить номер строки, в которой эта сумма максимальна.

Задание 5. Написать программу, которая вводит по строкам с клавиатуры двумерный массив и вычисляет сумму его элементов по столбцам

Задание 6. Дана целочисленная прямоугольная матрица. Определить: 1) количество строк, не содержащих ни одного нулевого элемента; 2) максимальное из чисел, встречающихся в заданной матрице более одного раза.

Задание 7. Написать программу, которая вводит по строкам с клавиатуры двумерный массив дробного типа (3*5 — три строки по пять элементов) и вычисляет среднее арифметическое элементов строк.

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: Работа со строками

Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с использованием строковых переменных

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Напишите программу, которая считывает по отдельности имя и фамилию, объединяет их в одной переменной и выводит на консоль.

```
#include <iostream>
#include <string>
#include <clocale>
int main(){
    setlocale(LC_ALL, "Russia");
    std::string name, surname, fullname;
    std::cout << "Введите ваше имя: ";
    std::cin >> name;
    std::cout << "Введите вашу фамилию: ";
    std::cin >> surname;
    fullname = name + " " + surname;
    std::cout << "Ваше имя: " << fullname << std::endl;
    return 0; }
```

Задание 2. Напишите программу, которая считывает с консоли две строки, проверяет их на равенство и выводит результат проверки на консоль.

```
#include <iostream>
#include <string>
#include <clocale>
int main(){
    setlocale(LC_ALL, "Russia");
    std::string s1, s2;
    std::cout << "Введите первую строку: ";
    std::cin >> s1;
    std::cout << "Введите вторую строку: ";
    std::cin >> s2;
    if(s1 == s2)
        std::cout << "Строка 1 совпадает со строкой 2" << std::endl;
    else
        std::cout << "Строка 1 не совпадает со строкой 2" << std::endl;
    return 0;
}
```

Задание 3. Дана строка, заканчивающаяся точкой. Определить, сколько слов в строке

Задание 4. Дана строка. Определить, сколько раз входит в нее группа букв abc

Задание 5. Дана строка. Преобразовать ее, удалив каждый символ * и повторив каждый символ, отличный от *.

Задание 6. Составить символьную строку из N звездочек

Задание 7. Написать программу, подсчитывающую количество цифр в заданной строке.

ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема: Работа с данными типа множество

Цель работы: организовывать программы и использованием структурированного типа данных «множество».

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Справочный материал:

В STL есть контейнер — set, он реализует такие сущности как множество и мультимножество. По сути это контейнеры, которые содержат некоторое количество отсортированных элементов. При добавлении нового элемента в множество он сразу становится на свое место так, чтобы не нарушать порядка сортировки. Потому как в множестве и мультимноестве все элементы сортируются автоматически. Множества содержат только уникальные элементы, а мультимножества могут содержать дубликаты, вот такая вот небольшая разница.

Для того, чтобы использовать множество или мультимножество необходимо подключить следующий заголовочный файл: #include <set>

Содержание работы:

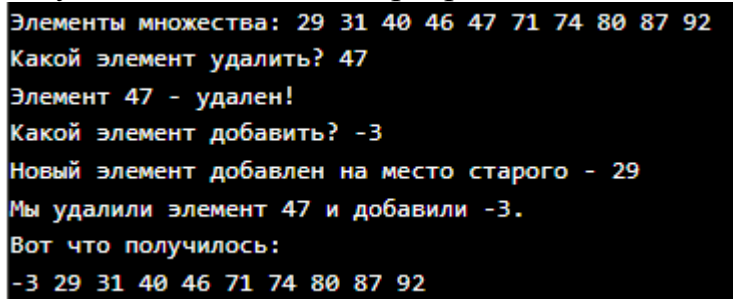
Задание 1. Написать программу с удалением и добавлением нового элемента.

```
#include <iostream>
#include <set> // заголовочный файл множеств и мультимножеств
#include <iterator>
#include <cstdlib>
using namespace std;
int main(){
    srand(time(NULL));
    set<int> mySet; // объявили пустое множество
    // добавляем элементы в множество
    for( int i = 0; i < 10; i++) {
        mySet.insert( rand() % 100 );
    }
    cout << "Элементы множества: ";
    copy( mySet.begin(), mySet.end(), ostream_iterator<int>(cout, " "));
    int del = 0;
    cout << "\nКакой элемент удалить? ";
    cin >> del;
    cout << "Элемент " << *mySet.find(del) << " - удален!" << endl;
    mySet.erase(del);
    int add = 0;
    cout << "Какой элемент добавить? ";
    cin >> add;
    cout << "Новый элемент добавлен на место старого - " <<
    *mySet.lower_bound(add) << endl;
    mySet.insert(add);
```

```
cout << "Мы удалили элемент " << del << " и добавили " << add << ".\nВот  
что получилось: " << endl;  
copy( mySet.begin(), mySet.end(), ostream_iterator<int>(cout, " "));  
return 0; }
```

Сначала удаляем элемент множества, это делается в строке 26, с помощью метода `erase()`. Потом добавляем новый элемент, строка 33. Кроме этого, обратите внимание на метод — `find()`, строка 25, он возвращает указатель на первый элемент множества, который равен его аргументу. Еще один интересный метод, которым воспользовались в программе называется — `lower_bound()`, строка 32. Метод `lower_bound()` возвращает указатель на первый элемент множества, значение которого больше либо равно аргументу.

Результат выполнения программы:



```
Элементы множества: 29 31 40 46 47 71 74 80 87 92  
Какой элемент удалить? 47  
Элемент 47 - удален!  
Какой элемент добавить? -3  
Новый элемент добавлен на место старого - 29  
Мы удалили элемент 47 и добавили -3.  
Вот что получилось:  
-3 29 31 40 46 71 74 80 87 92
```

Задание 2. В русском языке, когда говорят о количестве лет, то после числа используют слова "год", "года" и "лет". Например, 1 год, но 10 лет или 3 года. Требуется написать программу, которая в зависимости от числа добавляет правильное окончание.

Слово "год" добавляется ко всем числам, последняя (или единственная) цифра которых равна 1. "Года" для оканчивающихся на 2, 3, 4. Во всех остальных случаях используется слово "лет". Числа 11, 12, 13, 14 (или имеющие такой остаток от деления на 100) имеют "окончание" лет. Для того, чтобы определить, на какую цифру заканчивается число, надо найти остаток от его деления на 10. Для решения этой задачи удобно использовать множества.

Задание 3. Компьютер генерирует пять чисел в диапазоне от 1 до 15 включительно. Человек пытается их угадать. Программа должна выводить угаданные и неугаданные числа из тех, что сгенерировала программа, а также ошибочные числа пользователя.

ПРАКТИЧЕСКАЯ РАБОТА № 9

Тема: Составление программ с использованием текстовых файлов

Цель работы: научиться объявлять текстовые файлы в программе, научиться создавать и обрабатывать текстовые файлы с помощью языка программирования C++.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Справочный материал:

Текстовыми называются файлы, состоящие из любых символов. Они организуются по строкам, каждая из которых заканчивается символом «конца строки». Конец самого файла обозначается символом «конца файла». При записи информации в текстовый файл, просмотреть который можно с помощью любого текстового редактора, все данные преобразуются к символьному типу и хранятся в символьном виде.

Для работы с файлами используются специальные типы данных, называемые *потоками*. Поток **ifstream** служит для работы с файлами в режиме чтения, а **ofstream** в режиме записи. Для работы с файлами в режиме как записи, так и чтения служит поток **fstream**.

В программах на C++ при работе с текстовыми файлами необходимо подключать библиотеки **iostream** и **fstream**.

Для того чтобы записывать данные в текстовый файл, необходимо:

1. описать переменную типа **ofstream**.
2. открыть файл с помощью функции **open**.
3. вывести информацию в файл.
4. обязательно закрыть файл.

Для считывания данных из текстового файла, необходимо:

1. описать переменную типа **ifstream**.
2. открыть файл с помощью функции **open**.
3. считать информацию из файла, при считывании каждой порции данных необходимо проверять, достигнут ли конец файла.
4. закрыть файл.

Содержание работы:

Задание 1. Создать текстовый файл **D:\\sites\\accounts.txt** и записать в него **n** вещественных чисел

```
#include «stdafx.h»
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
int main(){
    setlocale (LC_ALL, «RUS»);
    int i, n;
    double a;
    //описывает поток для записи данных в файл ofstream f;
    //открываем файл в режиме записи,
```

```

//режим ios::out устанавливается по умолчанию
f.open(«D:\\sites\\accounts.txt», ios::out);
//вводим количество вещественных чисел
cout<<«n=»; cin>>n;
//цикл для ввода вещественных чисел и записи их в файл
for (i=0; i<n; i++){
cout<<«a=»;
//ввод числа
cin>>a;
f<<a<<«\t»;
//закрытие потока
f.close();
system(«pause»);
return 0;}

```

Задание 2. В текстовом файле D:\\game\\accounts.txt хранятся вещественные числа, вывести их на экран и вычислить их количество.

```

#include «stdafx.h»
#include <iostream>
#include <fstream>
#include <iomanip>
#include <stdlib.h>
using namespace std;
int main(){
setlocale (LC_ALL, «RUS»);
int n=0;
float a;
fstream F;
//открываем файл в режиме чтения
F.open(«D:\\sites\\accounts.txt»);
//если открытие файла прошло корректно, то
if (F){
//цикл для чтения значений из файла; выполнение цикла прервется,
//когда достигнем конца файла, в этом случае F.eof() вернет истину.
while (!F.eof()){
//чтение очередного значения из потока F в переменную a
F>>a;
//вывод значения переменной a на экран
cout<<a<<«\t»;
//увеличение количества считанных чисел
n++;}
//закрытие потока
F.close();
//вывод на экран количества считанных чисел
cout<<«n=»<<n<<endl;}

```

```
//если открытие файла прошло некорректно, то вывод
//сообщения об отсутствии такого файла
else cout<<» Файл не существует»<<endl;
system(«pause»);
return 0;}
```

Задание 3. Дан файл, содержащий текст, набранный строчными русскими буквами. Получить в другом файле тот же текст, записанный прописными буквами.

Задание 4. Дан файл, содержащий произвольный текст. Определить, чего в нем больше: русских букв или цифр.

Задание 5. Дан файл, содержащий текст на русском языке. Определить, входит ли заданное слово в указанный текст, и если входит, то сколько раз.

Задание 6. Написать программу в соответствии с вариантом

Номер варианта	Задание
1	Дан файл f, компоненты которого являются целыми числами. Записать в файл g, компоненты файла f, исключив повторные вхождения чисел.
2	Дан файл f, компоненты которого являются действительными числами. Найти: 1. наибольшее из значений компонентов f; 2. наименьшее из значений компонентов с четными номерами; 3. наибольшее из значений модулей компонентов с нечетными номерами; 4. сумму наибольшего и наименьшего из значений компонентов файла f; 5. разность первой и последней компоненты файла f.
3	Дан символьный файл f. Подсчитать число вхождений в файл каждой из букв a, b, c, d, e, f. Результат вывести в файл g в виде таблицы с комментариями.
4	Дан файл f, компоненты которого являются целыми числами. Записать в файл g все четные числа исходного файла, в файл h – все нечетные. Порядок следования чисел сохраняется. Записать в файл g и h комментарии.
5	Дан текстовый файл, содержащий программу на языке C. Проверить эту программу на соответствие числа открывающих и закрывающих фигурных скобок.
6	Дан символьный файл f. Найти и записать в файл g самое длинное слово файла f, снабдив его комментарием.
7	Дан файл f, компоненты которого являются целыми числами. Получить в файле g все компоненты файла f: 1. являющиеся четными числами; 2. делящиеся на 3 и не делящиеся на 7;

Номер варианта	Задание
	3. являющиеся точными квадратами. Записать в файл g комментарий.
8	Дан файл f. Создать два файла, записав в первый из них все четные числа, расположив их в порядке возрастания, а во второй – все нечетные, расположив их в порядке убывания.
9	Дан текстовый файл f. Переформатировать исходный файл, разделяя его на строки так, чтобы каждая строка содержала столько символов, сколько содержит самая короткая строка исходного файла.
10	Дан файл f. Создать два файла, записав в первый из них среднее геометрическое всех четных чисел, а во второй – среднее арифметическое всех нечетных чисел.
11	Дан числовой файл f. Выбрать все значения, которые делятся нацело на 2 и 4, но не делятся на 6. Записать эти значения в файл g, а все остальные – в файл h.
12	Дан текстовый файл f. Определить, являются ли первые два символа цифрами и если да, то четно ли это число. Записать его в файл g, если оно четно и в h, если оно нечетно.
13	Дан текстовый файл f. Создать новый файл g и переписать в него исходный файл в обратном порядке, разделив пробелами.
14	Дан текстовый файл f. Создать новый файл, включив в него только те строки исходного файла, которые содержат самый часто используемый символ исходного файла.
15	Дан текстовый файл, содержащий программу на языке C. Проверить эту программу на соответствие открывающих и закрывающих скобок разных типов - (), {}, [].

ПРАКТИЧЕСКАЯ РАБОТА № 10

Тема: Создание программ с использованием типизированных и нетипизированных файлов

Цель работы: научиться описывать в программе типизированные файлы, обрабатывать типизированные и нетипизированные файлы.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1.

1. Дан файл вещественных чисел a.txt. Найти количество отрицательных и количество положительных элементов.
2. Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и произведение элементов меньших 1 и больших 0.
3. Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму отрицательных элементов.
4. Дан файл вещественных чисел a.txt. Найти количество элементов равных 5 и сумму положительных элементов.
5. Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму положительных элементов.
6. Дан файл вещественных чисел a.txt. Найти количество положительных элементов и сумму положительных элементов.
7. Дан файл вещественных чисел a.txt. Найти количество отрицательных и количество положительных элементов.
8. Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и произведение элементов меньших 1 и больших 0.
9. Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму отрицательных элементов.
10. Дан файл вещественных чисел a.txt. Найти количество элементов равных 5 и сумму положительных элементов.
11. Дан файл вещественных чисел a.txt. Найти количество нулевых элементов и сумму положительных элементов.
12. Дан файл вещественных чисел a.txt. Найти количество положительных элементов и сумму положительных элементов.
13. Дан файл вещественных чисел a.txt. Найти количество положительных элементов и произведение элементов меньших 1 и больших 0.
14. Дан файл вещественных чисел a.txt. Переписать положительные элементы в файл b.txt

Задание 2.

1. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти сумму положительных элементов в двух файлов.
2. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов
3. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти количество нулевых элементов в двух файлов

4. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти сумму положительных элементов в двух файлов.
5. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов
6. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти количество нулевых элементов в двух файлов
7. Дан файл вещественных чисел a.txt. Переписать в файл a2.txt положительные элементы файла a(n) умноженные на 5.
8. Дан файл вещественных чисел a.txt . Переписать в файл a2.txt все ненулевые элементы файла a.txt
9. Дан файл вещественных чисел a.txt или a.txt. Переписать в файл a2.txt ненулевые элементы файла a.txt разделенные на 5.
10. Дан файл вещественных чисел a.txt. Переписать в файл a2.txt отрицательные элементы файла a.txt умноженные на 2.
11. Дано 2 файла вещественных чисел a1.txt и a2.txt. В каком из двух данных файлов больше отрицательных элементов?
12. В данном файле a2.txt каких элементов больше, равных 0 или равных 1?
13. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов
14. Дано 2 файла вещественных чисел a1.txt и a2.txt. Найти произведение отрицательных элементов двух файлов

Задание 3.

1. Организовать текстовый файл. Заменить в файле все маленькие латинские буквы на большие.(создавая новый дополнительный файл)
2. Из заданного входного файла считать символы и записать в один новый файл только буквы, в другой новый файл только цифры .
3. Организовать текстовый файл. Организовать замену символов в файле. "старый" символ и "новый" символ запрашиваются и вводятся с клавиатуры.(создавая новый дополнительный файл)
4. Организовать текстовый файл. Преобразовать файл, удалив в нем лишние пробелы.(создавая новый дополнительный файл)
5. Организовать текстовый файл, состоящий из строк. Заменить в файле все большие латинские буквы на маленькие . создавая новый дополнительный файл)
6. Организовать текстовый файл. Заменить в файле все цифры на '*'.(создавая новый дополнительный файл)
7. Организовать текстовый файл. Заменить в файле все буквы (нецифры) на '*'.(создавая новый дополнительный файл)
8. Организовать текстовый файл. Удалить в файле все цифры. (создавая новый дополнительный файл)
9. Из заданного входного файла считать символы и записать в один новый файл только большие латинские буквы, в другой новый файл только малые латинские буквы и посчитать количество цифр.

10. Организовать текстовый файл. Удалить в файле все буквы. (создавая новый дополнительный файл)

11. Из заданного входного файла считать символы и записать в новый файл все символы за исключением символов разделителей : пробелы , точки , запятые, двоеточия и т.д.

12. Организовать текстовый файл. Оставив в файле только буквы. (создавая новый дополнительный файл)

13. Из заданного входного файла считать символы и записать в новый файл только большие буквы латинского алфавита.

14. Организовать файл вещественных чисел. Заменить все положительные компоненты файла их квадратными корнями, а все отрицательные компоненты их квадратами, создавая новый дополнительный файл)

ПРАКТИЧЕСКАЯ РАБОТА № 11

Тема: Организация процедур и функций

Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с применением процедур и функций.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Создадим функцию `f()`, которая не имеет входных значений и не формирует результат. При вызове этой функции на экран выводится строка символов "С Новым Годом, ".

```
#include <iostream>
using namespace std;
void f ( ) //Описание функции.
cout << "С Новым Годом, "; }
int main ( ) {
f ( ); //Вызов функции.
cout <<"Студент!" << endl;
f ( ); //Вызов функции.
cout <<"Преподаватель!" << endl;
f ( ); //Вызов функции.
cout <<"Народ!" << endl; }
```

Результатом работы программы будут три строки:

```
С Новым Годом, Студент!
С Новым Годом, Преподаватель!
С Новым Годом, Народ!
```

Задание 2. Найти максимальную сумму элементов строк матрицы 3×5 с использованием функций.

```
#include <stdio.h>
const int m=3, n=5;
//функция ввода
void inparr(int a[m][n]) {
int i,j;
for (i=0;i<m;i++)
for (j=0;j<n;j++)
scanf("%d",&a[i][j]);}
//функция вывода
void outarr (int a[m][n]){
int i,j;
printf("Matrica:\n");
for (i=0; i<m; i++){
for (j=0; j<n; j++)
printf("%5d", a[i][j]);
printf("\n");}}
```

```

int processarr(int a[m][n]){
int i,j,s,max;
for(i=0;i<m;i++){
s=0;
for (j=0;j<n;j++)
s+=a[i][j];
if (i==0) max=s;
else if (max<s) max=s;}
return max;}
void main(){
int b[m][n];
inparr(b);
outarr(b);
printf("Maximalnaya summa stroki = %d", processarr(b));}

```

Задание 3.

1. Создать функцию, которая возвращает меньшее из двух данных чисел. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

2. Создать функцию, которая переводит время, заданное в минутах в секунды. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

3. Создать функцию, которая определяет периметр треугольника по трем его сторонам. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

4. Создать функцию, которая возвращает номер квадранта, в котором находится точка. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

5. Создать функцию, которая возвращает среднее арифметическое трех данных чисел. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

6. Создать функцию, которая определяет площадь круга по его радиусу. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

7. Создать функцию, которая возвращает остаток от деления двух натуральных чисел. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

8. Создать функцию, которая переводит радианы в градусы. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

9. Создать функцию, которая определяет длину отрезка по его координатам. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

10. Создать функцию, которая возвращает в долларах сумму, заданную в рублях. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

11. Создать функцию, которая возвращает большее из двух данных чисел. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

12. Создать функцию, которая определяет длину окружности по заданному радиусу. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

13. Создать функцию, которая переводит скорость из км/час в м/сек. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

14. Создать функцию, которая возвращает среднее геометрическое двух данных чисел. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

15. Создать функцию, которая возвращает в рублях сумму, заданную в долларах. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

16. Создать функцию, которая переводит градусы в радианы. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

17. Создать функцию, которая переводит время, заданное в секундах в минуты. Для создаваемой функции: подобрать имя; указать тип функции; выбрать имена и типы входных параметров; описать тело функции с обязательным оператором в конце; в главной программе вызвать созданную функцию два раза с различными входными данными. Вывести результаты в главной программе.

Задание 4.

1. Оформить функцию поиска количества отрицательных элементов массива. В главной программе дано 3 одномерных массива a,b,c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

2. Оформить функцию поиска количества положительных элементов массива. В главной программе дано 2 одномерных массива x,y длиной 20 элементов каждый и один массив z длиной 5 элементов.. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

3. Оформить функцию поиска количества элементов равных 5 . В главной программе дано 3 одномерных массива p,q,r длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

4. Оформить функцию поиска количества нулевых элементов массива. В главной программе дано 3 одномерных массива arr1,arr2,arr3 длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

5. Оформить функцию поиска количества элементов массива, больших заданного числа alfa. В главной программе дано 2 одномерных массива a,b

длиной 15 элементов каждый. Применить функцию для каждого из 2-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

6. Оформить функцию поиска суммы отрицательных элементов массива. В главной программе дано 2 одномерных массива x , y длиной 10 элементов каждый и один массив z длиной 5 элементов.. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

7. Оформить функцию поиска суммы элементов массива больших 5. В главной программе дано 2 одномерных массива a, b, c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

8. Оформить функцию поиска суммы элементов массива больших 1. В главной программе дано 2 одномерных массива $mass1$, $mass2$ длиной 10 элементов каждый и один массив z длиной 5 элементов. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

9. Оформить функцию поиска суммы элементов массива, больших заданного числа $alfa$. В главной программе дано 4 одномерных массива a, b, c, d длиной 10 элементов каждый. Применить функцию для каждого из 4-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

10. Оформить функцию поиска суммы элементов массива, больших заданного числа $alfa$ и меньшего заданного числа $beta$. В главной программе дано 3 одномерных массива a, b, c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

11. Оформить функцию поиска произведения положительных элементов массива. В главной программе дано 4 одномерных массива a, b, c, d длиной 10 элементов каждый. Применить функцию для каждого из 4-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

12. Оформить функцию поиска произведения отрицательных элементов массива. В главной программе дано 4 одномерных массива x, y, z, f длиной 10 элементов каждый. Применить функцию для каждого из 4-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

13. Оформить функцию поиска произведения элементов массива, больших заданного числа $alfa$ и меньшего заданного числа $beta$. В главной программе дано 2 одномерных массива a, b длиной 10 элементов каждый и один массив z длиной 5 элементов. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

14. Оформить функцию поиска среднего арифметического отрицательных элементов массива. В главной программе дано 3 одномерных массива a, b, c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

15. Оформить функцию поиска среднего арифметического положительных элементов массива. В главной программе дано 3 одномерных массива a, b, c длиной 10 элементов каждый. Применить функцию для каждого

из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

16. Оформить функцию поиска среднего геометрического элементов массива. В главной программе дано 3 одномерных массива a,b,c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

17. Оформить функцию поиска арифметического элементов массива меньшего заданного числа betta. В главной программе дано 3 одномерных массива a,b,c длиной 10 элементов каждый. Применить функцию для каждого из 3-х заданных массивов. (в функции не должно быть операторов ввода или вывода)

ПРАКТИЧЕСКАЯ РАБОТА № 12

Тема: Применение рекурсивных функций

Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с применением рекурсивных функций.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Содержание работы:

Задание 1. Переделаем программу нахождения факториала так, чтобы получить таблицу факториалов. Для этого объявим цикл for, в котором будем вызывать рекурсивную функцию.

```
// factorial.cpp: определяет точку входа для консольного приложения.
#include "stdafx.h"
#include using
namespace std;
unsigned long int factorial(unsigned long int); // прототип рекурсивной
функции
int i = 1; // инициализация глобальной переменной для подсчёта кол-ва
рекурсивных вызовов
unsigned long int result; // глобальная переменная для хранения
возвращаемого результата рекурсивной функцией
int main(int argc, char* argv[]) {
    int n; // локальная переменная для передачи введенного числа с
клавиатуры
    cout << "Enter n!: ";
    cin >> n;
    for (int k = 1; k <= n; k++ ) {
        cout << k << "!" << "=" << factorial(k) << endl; }
    // вызов рекурсивной функции
    system("pause");
    return 0; }
unsigned long int factorial(unsigned long int f) // рекурсивная функция для
нахождения n!
{
    if (f == 1 || f == 0) // базовое или частное решение
        return 1; // все мы знаем, что 1!=1 и 0!=1
    //cout << "Step\t"<< i <<<"Result= "<< result << endl;
    result=f*factorial(f-1); // функция вызывает саму себя
    return result; }
```

В строках 16 — 19 объявлен цикл, в котором вызывается рекурсивная функция. Всё ненужное в программе закомментировано. Запустив программу, нужно ввести значение, до которого необходимо вычислить факториалы.

Результат работы программы:

1!=1 2!=2 3!=6 4!=24 5!=120 6!=720 7!=5040 8!=40320 9!=362880
10!=3628800 11!=39916800 12!=479001600 13!=6227020800 14!=87178291200

Задание 2. Ханойские башни. Даны три стержня, на один из которых нанизаны N колец, причем кольца отличаются размером и лежат в порядке уменьшения диаметра (меньшее на большем). Задача состоит в том, чтобы перенести пирамиду из n колец на третий стержень. Вторым стержнем можно пользоваться для временного размещения колец. Надо решить задачу за наименьшее число ходов. За один ход разрешается переносить только одно кольцо, причем нельзя класть большее кольцо на меньшее.

Задание 3.

1. Описать рекурсивную функцию для вычисления n –го члена ряда 3, 0.3, 0.03, 0.003, ... 0.00003
2. Описать рекурсивную функцию для вычисления n –го члена ряда 10, 5, $5/2$, $5/4$, $5/8$, $5/16$, ... $5/256$
3. Описать рекурсивную функцию для вычисления n –го члена ряда 6, 3, $3/2$, $3/4$, $3/8$, ... $3/256$
4. Описать рекурсивную функцию для вычисления n –го члена ряда 1, 2, 4, 8,256
5. Описать рекурсивную функцию для вычисления n –го члена ряда 2, 4, 6,20
6. Описать рекурсивную функцию для вычисления n –го члена ряда 1, 3, 5,21
7. Описать рекурсивную функцию для вычисления n –го члена ряда 1, 1.2, 1.4, 1.6, ... 3.0
8. Описать рекурсивную функцию для вычисления n –го члена ряда 0.7 ; 0.07 ; 0.007 ; 0.0007; ...0.00000000007
9. Описать рекурсивную функцию для вычисления n –го члена ряда 1, 3, 9, 27, 81729
10. Описать рекурсивную функцию для вычисления n –го члена ряда 7, 17, 27, 37, 97
11. Описать рекурсивную функцию для вычисления n –го члена ряда 3, 3.3, 3.6, 3.9, 4.2,6
12. Описать рекурсивную функцию для вычисления n –го члена ряда 20, 10, 5, $5/2$, $5/4$, $5/8$, $5/16$, ... $5/128$
13. Описать рекурсивную функцию для вычисления n –го члена ряда 5,2; 5,4; 5,6; 5,8; 6; 6,2; ... 10
14. Описать рекурсивную функцию для вычисления n –го члена ряда 40, 20, 10, 5, $5/2$, $5/4$, ... $5/128$
15. Описать рекурсивную функцию для вычисления n –го члена ряда 1, 4, 16, 64, 256, 1024, 4096

ПРАКТИЧЕСКАЯ РАБОТА № 13

Тема: Программирование модуля

Цель работы: научиться применять в программах стандартные библиотеки подпрограмм, создавать собственные библиотеки подпрограмм.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Справочный материал:

Модуль логически состоит из двух файлов - файла с исходным кодом (source file) и заголовочного файла (header file):

1. файл с исходным кодом (module.cpp) включает в себя определения функций, а также определения глобальных переменных и констант (если они есть);

в первой строчке обычно подключается заголовочный файл того же модуля: `#include "module.h"`

2. заголовочный файл (module.h) включает в себя прототипы функций, определения констант,

объявления глобальных переменных - но только для тех элементов модуля, о которых должны знать другие модули

3. Если какой-либо модуль использует данный, необходимо подключить его заголовочный файл (в двойных кавычках – это указывает на то, что заголовочный файл не системный, а собственный): `#include "module.h"`

4. Главный модуль программы обычно содержит только функцию `main` и не имеет заголовочного файла (поскольку функция `main` не используется в других модулях)

5. Некоторые модули включают только заголовочный файл – например, содержащий определения глобальных констант

Содержание работы:

Задание 1. Главный файл `main.cpp`

```
#include "file.h"
#include "search.h"
#include <iostream>
#include <locale.h>
using namespace std;
int main(int argc, char** argv) {
    setlocale(LC_ALL, "Russian");
    if (argc < 2){
        cout<<"Запуск: Triangles.exe inf.txt outf.txt"<<endl;
        return -1;}
    const char* inFileName = argv[1]; // Имя входного файла
    const char* outFileName = argv[2]; // Имя выходного файла
    int pointNum = countPoints(inFileName);
    if (pointNum < 0) {
        cout<<"Входной файл не существует"<<endl;
        return -2; }
    else if (pointNum < 4){
        cout<<"Входной файл слишком мал"<<endl;
```

```

return -3;}
double* px = new double[pointNum]; // точки
double* py = new double[pointNum]; // точки
if (!readPoints(inFileName, px, py, pointNum)){
cout<<"Ошибка при вводе точек из файла"<<endl;
return -3;}
const int maxTrNum = 3;
// Треугольники
double trX[maxTrNum][3], trY[maxTrNum][3];
searchLargestTriangles(px, py, pointNum,
trX, trY, maxTrNum);
if (!writeTriangles(outFileName, trX, trY, maxTrNum)){
cout<<"Не удалось записать результат в файл"<<endl;
return -4;}
cout<<"Программа успешно завершена"<<endl;
delete[] px;
delete[] py;
return 0;}

```

Задание 2. Разработать модуль, содержащий подпрограмму суммирования элементов массива.

Разбиваем текст программы на две части: подпрограмму размещаем в модуле, а тестирующую программу оставляем в качестве основной программы. Так как все структурные типы параметров должны быть предварительно объявлены, описываем тип массива в модуле.

ПРАКТИЧЕСКАЯ РАБОТА № 14

Тема: Использование указателей для организации связанных списков

Цель работы: научиться применять в программе динамические структуры данных.

Оборудование: ПК, программное обеспечение – Visual Studio, инструкции по выполнению работы.

Справочный материал:

Список связный потому что все узлы списка связаны между собой с помощью указателей, а динамический потому что динамически во время выполнения программы можно расширять данную структуру путем добавления новых узлов в список. В отличие от массива будь то статического, либо динамического, динамический список можно увеличивать во время работы программы.

В простейшем случае каждый узел содержит всего одну ссылку. Для определенности будем читать, что решается задача частотного анализа текста – определения всех слов, встречающихся в тексте и их количества. В этом случае область данных элемента включает строку (длиной не более 40 символов) и целое число. Каждый элемент содержит также ссылку на следующий за ним элемент. У последнего в списке элемента поле ссылки содержит NULL. Чтобы не потерять список, мы должны где-то (в переменной) хранить адрес его первого узла – он называется «головой» списка. В программе надо объявить два новых типа данных – узел списка Node и указатель на него PNode. Узел представляет собой структуру, которая содержит три поля - строку, целое число и указатель на такой же узел.

Содержание работы:

Задание 1. Написать программу, которая обрабатывает файл input.txt и составляет для него алфавитно-частотный словарь в файле output.txt.

```
void main() {
    PNode Head = NULL, p, where;
    FILE *in, *out;
    char word[80];
    int n;
    in = fopen ( "input.dat", "r" );
    while ( 1 ) { n = fscanf ( in, "%s", word ); // читаем слово из файла
        if ( n <= 0 ) break;
        p = Find ( Head, word ); // ищем слово в списке
        if ( p != NULL ) // если нашли слово,
            p->count ++; // увеличить счетчик
        else { p = CreateNode ( word ); // создаем новый узел
            where = FindPlace ( Head, word ); // ищем место
            if ( ! where ) AddLast ( Head, p );
            else AddBefore ( Head, where, p ); } }
    fclose(in);
    out = fopen ( "output.dat", "w" );
    p = Head;
```

```
while ( p ) { // проход по списку и вывод результатов
    fprintf ( out, "%-20s\t%d\n", p->word, p->count );
    p = p->next; }
fclose(out); }
```

В переменной *p* хранится значение, которое вернула функция *fscanf* (количество удачно прочитанных элементов). Если это число меньше единицы (чтение прошло неудачно или закончились данные в файле), происходит выход из цикла *while*.

Сначала пытаемся искать это слово в списке с помощью функции *Find*. Если нашли – просто увеличиваем счетчик найденного узла. Если слово встретилось впервые, в памяти создается новый узел и заполняется данными. Затем с помощью функции *FindPlace* определяем, перед каким узлом списка надо его добавить.

Когда список готов, открываем файл для вывода и, используя стандартный проход по списку, выводим найденные слова и значения счетчиков.

Задание 2. Вводится определенное число. Проверить, есть ли такой элемент на диагонали матрицы (двумерного массива).

Задание 3. В соответствии с вариантом разработать программу, которая содержит динамическую информацию в виде динамического односвязного списка.

Вариант 1. Составить программу, которая содержит динамическую информацию о наличии автобусов в автобусном парке. Сведения о каждом автобусе включают:

- номер автобуса;
- фамилию и инициалы водителя;
- номер маршрута.

Программа должна обеспечивать:

- начальное формирование данных обо всех автобусах в парке в виде односвязного списка;

- вывод всех автобусов;
- добавление автобуса в начало списка;
- добавление автобуса перед определенным автобусом;
- по запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

- при выезде каждого автобуса из парка вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;

- при въезде каждого автобуса в парк вводится номер автобуса, и программа удаляет данные об этом автобусе из списка автобусов, находящихся

на маршруте, и записывает эти данные в список автобусов, находящихся в парке;

Вариант 2. Составить программу, которая содержит текущую информацию о книгах в библиотеке. Сведения о книгах включают:

- номер УДК;
- фамилию и инициалы автора;
- название; • год издания;
- количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать:

• начальное формирование данных обо всех книгах в библиотеке в виде односвязного списка;

- добавление данных о книгах, вновь поступающих в библиотеку;
- удаление данных о списываемых книгах;
- добавление книги в начало списка;
- добавление книги в конец списка;

• добавление книги в позицию, отсортированную по имени в алфавитном порядке;

• по запросу выдаются сведения о наличии книг в библиотеке, упорядоченные по годам издания.

Вариант 3. Составить программу, которая содержит текущую информацию о заявках на авиабилеты. Каждая заявка включает:

- пункт назначения;
- номер рейса;
- фамилию и инициалы пассажира;
- желаемую дату вылета.

Программа должна обеспечивать:

- хранение всех заявок в виде односвязного списка;
- добавление заявок в список;
- удаление заявок;
- вывод заявок по заданному номеру рейса и дате вылета;
- вывод всех заявок.

Вариант 4. Составить программу, которая содержит текущую информацию о книгах в библиотеке. Сведения о книгах включают:

- номер УДК;
- фамилию и инициалы автора;
- название;
- год издания;
- количество экземпляров данной книги в библиотеке.

Программа должна обеспечивать:

• начальное формирование данных обо всех книгах в библиотеке в виде односвязного списка;

- добавление книги, отсортированной по автору;
- добавление книги перед указанной книгой;

- добавление книги после указанной книги.
- удаление выбранной книги.
- при выдаче каждой книги на руки вводится номер УДК, и программа уменьшает значение количества книг на единицу или выдает сообщение о том, что требуемой книги в библиотеке нет или требуемая книга находится на руках;
- при возвращении каждой книги вводится номер УДК, и программа увеличивает значение количества книг на единицу;
- по запросу выдаются сведения о наличии книг в библиотеке.

Вариант 5. Картотека в бюро обмена квартир организована в виде односвязного списка. Сведения о каждой квартире включают:

- количество комнат;
- этаж;
- площадь;
- адрес.

Написать программу, которая обеспечивает:

- начальное формирование картотеки;
- ввод заявки на обмен;
- поиск в картотеке подходящего варианта: при равенстве количества комнат и этажа и различии площадей в пределах 10% соответствующая карточка выводится и удаляется из списка, в противном случае поступившая заявка включается в список;
- вывод всего списка.
- удаление квартиры по адресу.
- добавление новой квартиры перед указанной в список.
- добавление новой квартиры после указанной.
- добавление квартиры отсортированной по адресу.

Программа должна обеспечивать диалог с помощью меню и контроль ошибок при вводе.

ПРАКТИЧЕСКАЯ РАБОТА № 15

Тема: Создание проекта с использованием компонентов для работы с текстом

Цель работы: Обобщить знания по управляющим элементам Delphi; получить практические навыки работы с кнопочными компонентами, овладеть практическими навыками в организации ввода/вывода и обработки значений, получить практические навыки создания приложений

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Даны две целочисленные матрицы. Найти сумму этих матриц

1. Запустите Delphi выполнив команду Пуск/Программы/Borland Delphi 7/Delphi 7

2. Поместите на форму необходимые компоненты:

- 5 компонентов Label- для вывода информационных сообщений «Число строк», «Число столбцов», «Матрица А», «Матрица В», «Матрица $C=A+B$ »
- 2 компонента Edit- для ввода числа строк и числа столбцов
- 3 компонента StringGrid- для ввода значений матриц А, В и С
- компонент Button1- кнопка, предназначенная для выдачи команд на ввод количества строк и столбцов, а также установку размерности компонентов StringGrid
- компонент Button2- кнопка, предназначенная для выдачи команд для подсчета суммы двух матриц и вывода значений матрицы С в компонент StringGrid

3. Набрать текст модуля:

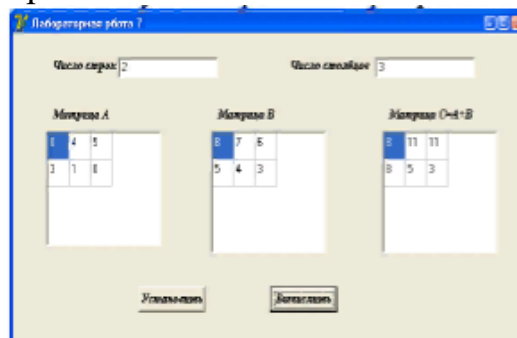
```
unit Unit1;  
interface  
uses Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,  
Forms, Dialogs, StdCtrls, Grids;  
type TForm1 = class(TForm)  
StringGrid1: TStringGrid;  
StringGrid2: TStringGrid;  
StringGrid3: TStringGrid;  
Edit1: TEdit;  
Edit2: TEdit;  
Button1: TButton;  
Button2: TButton;  
Label1: TLabel;  
Label2: TLabel;  
Label3: TLabel;  
Label4: TLabel;  
Label5: TLabel;  
procedure Button1Click(Sender: TObject);  
procedure Button2Click(Sender: TObject);  
end;  
implementation
```

```

private { Private declarations }
public { Public declarations }
end;
var Form1: TForm1; n,m: integer;
implementation {$R *.dfm}
procedure TForm1.Button1Click(Sender:
TObject);
begin
n:= StrToInt(Edit1.Text);
m:= StrToInt(Edit2.Text);
StringGrid1.ColCount:=m;
StringGrid1.RowCount:=n;
StringGrid2.ColCount:=m;
StringGrid2.RowCount:=n;
StringGrid3.ColCount:=m;
StringGrid3.RowCount:=n;
end;
procedure TForm1.Button2Click(Sender:
TObject);
var a,b,c: array [1..10, 1..10] of integer; I,j: integer;
Begin
// ввод значений исходных матриц A и B
For I:=1 to n do
for j:=1 to m do begin
a[I,j]:= StrToInt(StringGrid1.Cells[j-1,I-1]);
b[I,j]:= StrToInt(StringGrid2.Cells[j-1,I-1]);
end;
// Вычисление значений матрицы C=A+B
For I:=1 to n do
for j:=1 to m do c[I,j]:=a[I,j]+b[i,j];
// вывод значений матрицы C
For I:=1 to n do
for j:=1 to m do
StringGrid3.Cells[j-1,I-1]:=IntToStr(c[I,j]);
end.

```

4. Результат работы программы:



5. Сохранить проект.

Задание 2. Разработать приложение для вычисления стоимости покупок.

Ход работы:

В качестве примера на рис. 1 приведена совокупность алгоритмов программы **Стоимость покупки**, а на рис. 2 – ее диалоговое окно. После разработки диалогового окна и алгоритмов обработки событий можно приступить к написанию программы.

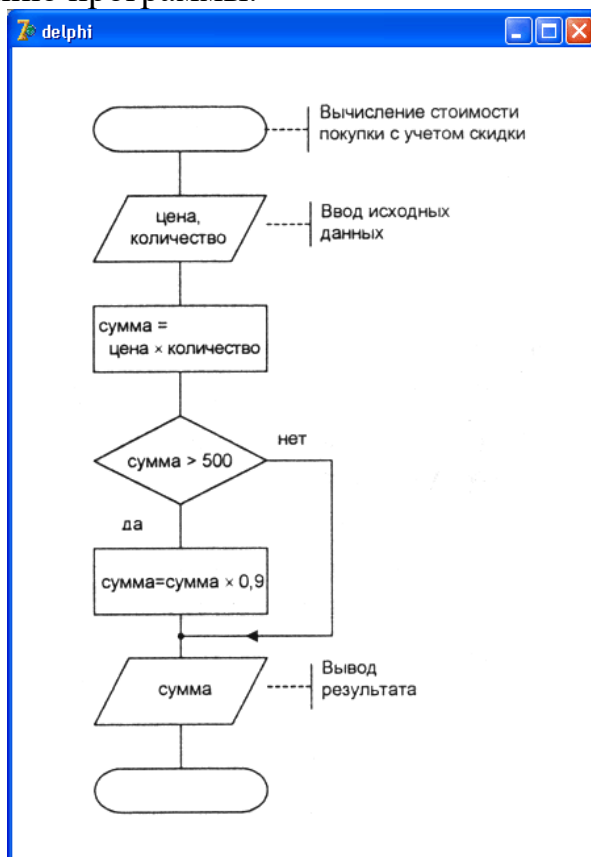


Рис. 1. Алгоритм программы вычисления стоимости покупки – совокупность алгоритмов обработки событий на компонентах формы

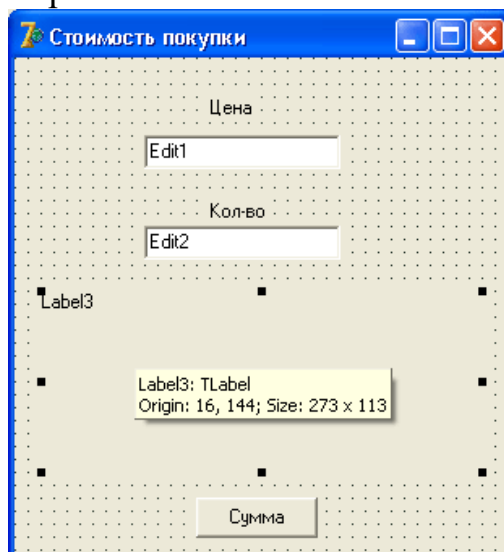


Рис. 2. Окно (форма) программы Стоимость покупки

1. Запустите Delphi выполнив команду Пуск/Программы/Borland Delphi 7/Delphi 7
2. Поместите на форму необходимые компоненты:

3.Набрать текст модуля:

```
unit pokupka_1;
interface uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;
type
TForm1 = class(TForm)
Edit1: TEdit;
Edit2: TEdit;
Label1: TLabel;
Label2: TLabel;
Button1: TButton;
Label3: TLabel;
procedure Button1Click(Sender: TObject);
procedure Edit2KeyPress(Sender: TObject;
var Key: Char);
procedure Edit1KeyPress(Sender: TObject;
var Key: Char);
private { Private declarations }
public { Public declarations }
end;
var Form1: TForm1;
implementation
{$R *.dfm}
// подпрограмма
procedure Summa;
var cena: real; // цена
kol: integer; // количество
s: real; // сумма
mes: string[255]; // сообщение
begin
cena := StrToFloat(Form1.Edit1.Text);
kol := StrToInt(Form1.Edit2.Text);
s := cena * kol;
if s > 500 then
begin
s := s * 0.9;
mes := 'Предоставляется скидка 10%' + #13;
end;
mes := mes+ 'Стоимость покупки: ' + FloatToStrF(s,ffFixed,4.2) + ' руб.';
Form1.Label3.Caption := mes;
end;
// щелчок на кнопке Стоимость
procedure TForm1.Button1Click(Sender: TObject);
begin
```

```

Summa; // вычислить сумму покупки
end;
// нажатие клавиши в поле Количество
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
case Key of
'0'..'9',#8;; // цифры и клавиша <Backspace>
#13: Summa; // вычислить стоимость покупки
else Key: = Chr(0); // символ не отображать
end;
end;
// нажатие клавиши в поле Цена
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
case Key of
'0'..'9', #8;; // цифры и клавиша <Backspace>
#13: Form1.Edit2.SetFocus; // клавиша Enter
begin
if Key = '.'
then Key: = ', if Pos(',', Edit1.Text) <> 0
then Key: = Chr(0);
end;
else // все остальные символы запрещены
Key: = Chr(0);
end; end; end.
5.Сохранить проект.

```

ПРАКТИЧЕСКАЯ РАБОТА № 16

Тема: Создание проекта с использованием компонентов для работы с текстом

Цель работы: Обобщить знания по управляющим элементам Delphi; получить практические навыки работы с кнопочными компонентами, овладеть практическими навыками в организации ввода/вывода и обработки значений, получить практические навыки создания приложений

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Написать программу *Мили-километры*, которая пересчитывает расстояние из миль в километры (1 миля равна 1 км 609,34 м). Рекомендуемый вид формы приведен на рис. 1.

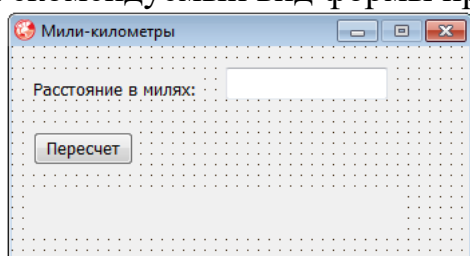


Рис. 1. Форма программы

Задание 2. Написать программу *Конвертор*, которая пересчитывает цену из долларов в рубли. Рекомендуемый вид формы приведен на рис. 2. Программа должна быть спроектирована таким образом, чтобы пользователь мог ввести в поля редактирования только правильные данные (дробные числа). При нажатии клавиши *Enter* в поле *Курс* курсор должен переходить в поле *Цена*, а при нажатии этой же клавиши в поле *Цена* — на кнопку *Пересчет*.

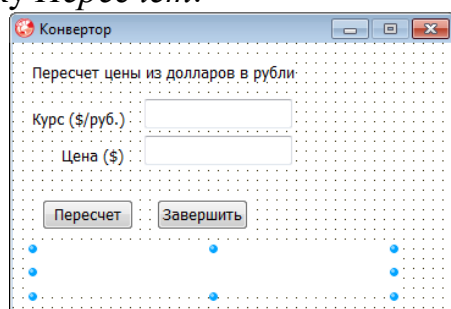


Рис. 2. Форма программы

Задание 3. Написать программу, которая вычисляет силу тока в электрической цепи. Рекомендуемый вид формы приведен на рис. 3. Программа должна быть спроектирована таким образом, чтобы кнопка *Вычислить* была доступна только в том случае, если пользователь ввел величину напряжения и сопротивления.

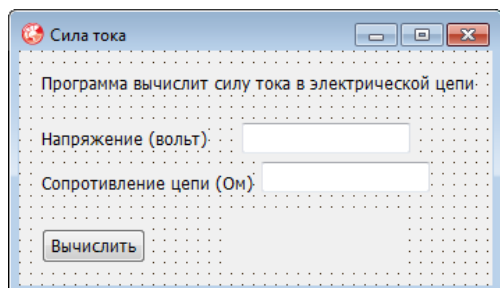


Рис. 3. Форма программы Сила тока

ПРАКТИЧЕСКАЯ РАБОТА № 17

Тема: Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени

Цель работы: Цель работы: сформировать навыки разработки приложений с использованием компонент ввода и отображения чисел, дат и времени в среде визуального программирования Delphi, изучить особенности их использования

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

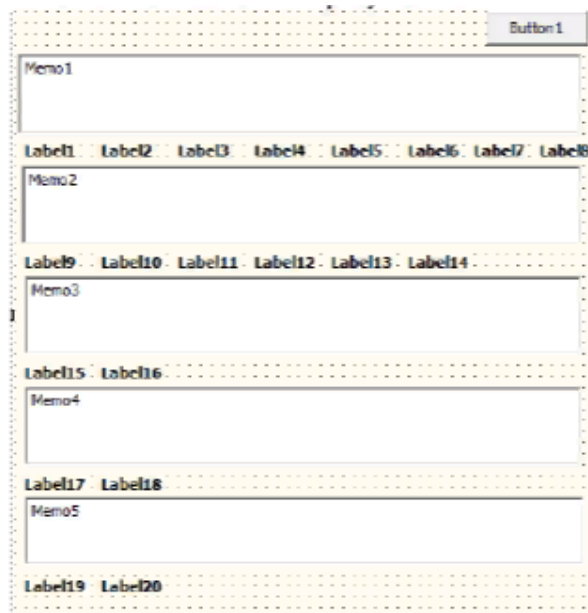
Задание 1. Разработать программу, демонстрирующую действие процедур и функций, оперирующих с системными значениями даты и времени.
1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.

2.Создайте в папке своей группы новую папку и назовите её Date1. Сохраните проект в созданную ранее папку, выполнив команду File - Save Project as...

3.Задайте для формы следующие свойства:

Свойство	Значение
BorderStyle	bsNone
ClientHeight	441
ClientWidth	423
Color	clCream
Left	102
Position	poScreenCenter
Top	98

4.Разместить на форме компоненты:



5.Задайте для элементов управления Label значение свойства Caption в соответствии с рисунком. Для этого выделите элемент управления и перейдите в Object Inspector на страницу Properties (свойства).

6. Задайте для элементов управления Мемо значение свойства Lines в соответствии с таблицей. Для этого выделите элемент управления и перейдите в Object Inspector на страницу Properties (свойства).

	Значение
Мемо1	DecodeTime (Time, hr, min, sec, msec) процедура, преобразующая текущее значение времени в формат целых чисел по соответствующим переменным.
Мемо2	DecodeDate (Date, year, mon, day) процедура, преобразующая текущее значение даты в формат целых чисел по соответствующим переменным.
Мемо3	DateToStr (Date) – функция, возвращающая строку с текущим значением даты в формате day.mon.year
Мемо4	TimeToStr (Time) – функция, возвращающая строку с текущим значением времени в формате hr:min:sec
Мемо5	DayOfWeek (Date) – функция, возвращающая номер дня недели по следующему соответствию: 1 – воскресенье, 2 – понедельник, 3 – вторник, 4 – среда, 5 – четверг, 6 – пятница, 7 – суббота.

7. Задайте для элемента управления Button1 значение свойства Caption равным Х.

8. Введите в окне кода процедуру для отображения даты и времени календаря компьютера.

```

procedure TForm1.FormCreate(Sender: TObject);
var hr,min,sec,msec:word;
    year,mon,day:word;

begin
    DecodeTime(Time, hr, min, sec, msec);
    Label2.Caption:=IntToStr(hr);
    Label4.Caption:=IntToStr(min);
    Label6.Caption:=IntToStr(sec);
    Label8.Caption:=IntToStr(msec);

    DecodeDate(Date, year, mon, day);
    Label10.Caption:=IntToStr(year);
    Label12.Caption:=IntToStr(mon);
    Label14.Caption:=IntToStr(day);

    Label16.Caption:=DateToStr(Date);
    Label18.Caption:=TimeToStr(Time);
    Label20.Caption:=IntToStr(DayOfWeek(Date));

end;

```

9. Запишите для командной кнопки Button1 процедуру, которая по щелчку на эту кнопку закрывает форму.

10. Сохраните изменения и запустите проект, убедитесь в его работоспособности.

The screenshot shows a Delphi application window with a yellow background. It contains several text boxes and labels displaying the results of various time and date conversion functions. The functions and their results are as follows:

- DecodeTime** (Time, hr, min, sec, msec) процедура, преобразующая текущее значение времени в формат целых чисел по соответствующим переменным.
часы: 14 минуты: 12 секунды: 21 миллисекунды: 649
- DecodeDate** (Date, year, month, day) процедура, преобразующая текущее значение даты в формат целых чисел по соответствующим переменным.
год: 2013 месяц: 4 день: 22
- DateToStr** (Date) - функция, возвращающая строку с текущим значением даты в формате day.month.year
дата: 22.04.2013
- TimeToStr** (Time) - функция, возвращающая строку с текущим значением времени в формате hh:mm:ss
время: 14:12:21
- DayOfWeek** (Date) - функция, возвращающая номер дня недели по следующему соответствию: 1 - воскресенье, 2 - понедельник, 3 - вторник, 4 - среда, 5 - четверг, 6 - пятница, 7 - суббота.
номер дня недели: 2

Задание 2. Разработать приложение, с помощью которого пользователь в одностраничном блокноте выбирает одну из представленных закладок. На рабочем поле блокнота высвечивается соответствующая надпись: год, месяц или день календаря компьютера.

Компонент приложения, содержащий несколько страниц, каждая из которых имеет ярлычок в виде закладки, называется элементом с закладками. Страницы пользователь может выбирать, щелкая по закладкам (корешкам или ярлычкам).

1. В Delphi на странице Win32 Палитры компонентов расположены две составляющие, работающие на программирование элементов с закладками: **TabControl** – одностраничный блокнот; и **PageControl** – многостраничный блокнот. Данные компоненты являются контейнерами и могут содержать в себе другие элементы или группы. **TabControl** имеет несколько стилей отображения (свойство **Style**):


- **tsTabs** – стандартные закладки объемного вида;
- **tsButtons** – закладки в виде кнопок;
- **tsFlatButtons** – закладки в виде плоских кнопок.

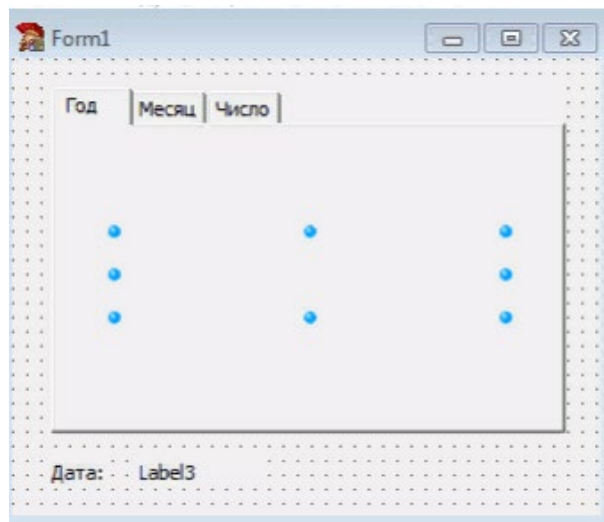
Свойство **Tabs** задает число и названия закладок блокнота.

Свойство **TabIndex** указывает текущую закладку в массиве **Tabs**. Программист может использовать данное свойство для переключения на нужную закладку блокнота. Нумерация начинается с 0. Если значение **TabIndex** равно – 1, то ни одна закладка не выбрана.

Свойство **HotTrack** задает подсвечивание названия закладки при наведении на неё курсора мыши.

2. Создайте новый проект.

- 3.Создайте в папке своей группы новую папку и назовите её Date2. Сохраните проект в созданную ранее папку, выполнив команду File - Save Project as...
- 4.Разместите на форме компоненты Label, компонент TabControl. В Label1 будет размещаться надпись блокнота, в Label2, Label3 отображаются текущая дата календаря компьютера.
- 5.Задайте для элементов управления значение свойства Caption в соответствии с рисунком. Для этого выделите элемент управления и перейдите в Object Inspector на страницу Properties (свойства).
- 6.Выделите элемент управления TabControl1, в Object Inspector перейдите на страницу Properties (свойства) и найдите свойство Tabs, щелкнуть на кнопке . В открывшемся окне String List Editor введите 3 строки со значениями год, месяц и число соответственно.



- 7.Перейдите в окно редактирования кода и опишите в разделе VAR переменные y, m, d типа word как глобальные переменные модуля для хранения значений даты календаря компьютера.

```
var
    Form1: TForm1;
    y, m, d: word;
```

- 8.Введите в окне редактора кода процедуру для отображения значений даты календаря компьютера:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    Label3.Caption:= DateToStr(Date); //отображение текущей даты
    //календаря компьютера
    decodeDate(Date, y, m, d);
    Label1.Caption:=intToStr(y); //надпись блокнота
end;
```

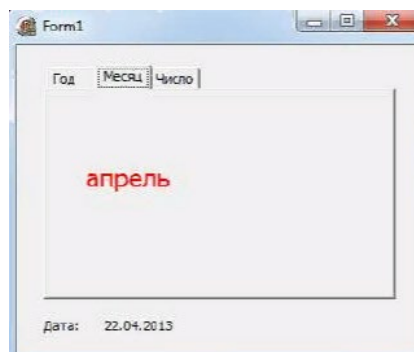
- 9.Выделите элемент управления TabControl1, в Object Inspector перейдите на страницу Events (события) и найдите событие OnChange.
- 10.Введите в окне редактора кода процедуру для работы с элементом управления TabControl1.

```

procedure TForm1.TabControl1Change(Sender: TObject);
begin
case tabcontrol1.TabIndex of
0:
begin
label1.Caption:='';label1.Caption:=IntToStr(y);
end;
1:
begin
label1.Caption:='';
case m of
1: label1.Caption:='январь';
2: label1.Caption:='февраль';
3: label1.Caption:='март';
4: label1.Caption:='апрель';
5: label1.Caption:='май';
6: label1.Caption:='июнь';
7: label1.Caption:='июль';
8: label1.Caption:='август';
9: label1.Caption:='сентябрь';
10: label1.Caption:='октябрь';
11: label1.Caption:='ноябрь';
12: label1.Caption:='декабрь';
end;
end;
2:
begin
label1.Caption:='';
label1.Caption:=IntToStr(d);
end;
end;
end;
end;

```

11.Сохраните изменения и запустите проект, убедитесь в его работоспособности.



ПРАКТИЧЕСКАЯ РАБОТА № 18

Тема: Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени

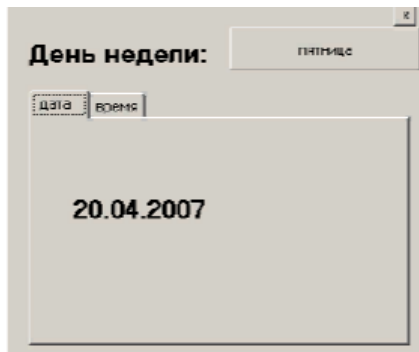
Цель работы: Цель работы: сформировать навыки разработки приложений с использованием компонент ввода и отображения чисел, дат и времени в среде визуального программирования Delphi, изучить особенности их использования

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработайте приложение, с помощью которого пользователь в одностраничном блокноте выбирает одну из представленных вкладок. На рабочем поле блокнота высвечивается соответствующая надпись: дата, время календаря компьютера. На панели отображается день недели.

Задание 2. Измените, проект так, чтобы каждый день недели был раскрашен в разный цвет.




ПРАКТИЧЕСКАЯ РАБОТА № 19

Тема: Создание проекта с использованием полос прокрутки для ввода информации.

Цель работы: сформировать умения использования полос прокрутки для ввода информации Delphi, изучить основные свойства; сформировать умения по созданию процедур на основе событий компонента.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Справочный материал:

Полоса прокрутки – классический элемент оконного интерфейса, использующийся для скроллинга информации, которая не помещается целиком в область вывода. В объектах просмотра и редактирования Windows этот элемент появляется при необходимости автоматически. В палитре VCL-компонент для создания полос прокрутки как самостоятельных элементов введен отдельный компонент : *ScrollBar* – самостоятельная полоса прокрутки, может быть горизонтальной или вертикальной, что определяется свойством *Kind=sbHorizontal/sbVertical*.

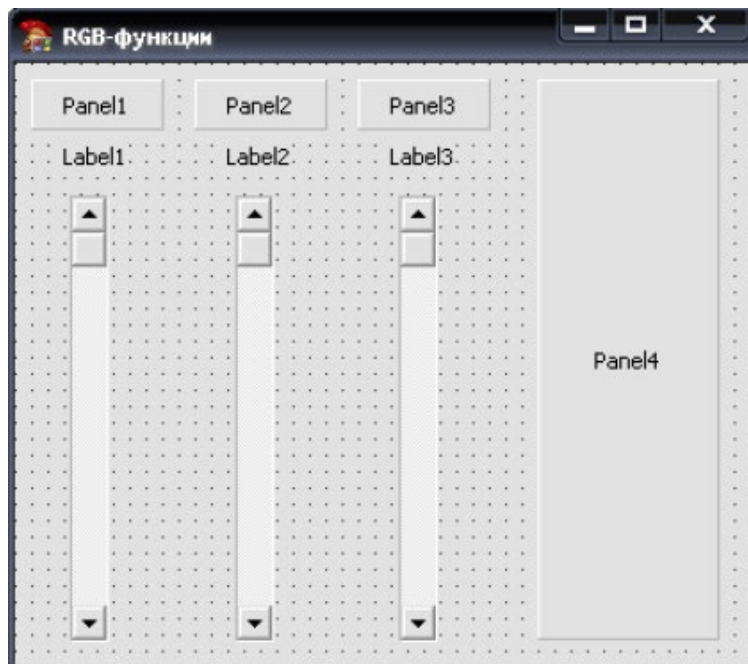
Свойствами *Min* и *Max* можно задать края диапазона целых значений, соответствующих положению бегунка полосы (свойство *Position*). Обычно также выбирают размеры скачков бегунка при щелчке на стрелках или на свободной полосе, определяя свойства *SmallChange* и *LargeChange*.

При перемещении бегунка *ScrollBar* генерируется событие *OnChange* (при попытке перемещения – *OnScroll*). При этом можно получать и обрабатывать данные как свойство *Position*. Во время работы программы значения свойств *Position*, *Min*, *Max* можно задавать с помощью метода *SetParams*.

Содержание работы:

Задание 1. Разработать проект демонстрации работы RGB – функций (установок цвета по трем составляющим) с помощью полос прокрутки. Каждый бегунок полос прокрутки должен будет менять вклад RGB –компонента, отображающийся на панели как цвет, а на метке как число. Результирующий цвет должен отображаться на панели.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *ScrollBox1*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместите на форме компоненты в соответствии с рисунком.



4. Задайте для элемента управления *ScrollBar1* значение свойства *Name=RedBar* и установите значение свойств *Max=255*, *Position=122*.
5. Задайте для элемента управления *Scrollbar2* значение свойства *Name=GreenBar* и установите значение свойств *Max=255*, *Position=122*.
6. Задайте для элемента управления *Scrollbar3* значение свойства *Name=BlueBar* и установите значение свойств *Max=255*, *Position=122*.

7. Выделите элемент *RedBar*, в *Object Inspector* перейдите на вкладку *Events*. Найдите событие *OnChange*, справа от него в поле сделайте двойной щелчок левой кнопкой мыши. Оказавшись в коде программы, введите следующий код:

```
procedure TForm1.RedBarChange(Sender: TObject);
begin
  Panel1.Color:=TColorRef(RGB(RedBar.Position,0,0));
  Label1.Caption:=IntToStr(RedBar.Position);
  Panel4.Color:=TColorRef(RGB(RedBar.Position,GreenBar.Position,BlueBar.Position))
end;
```

Функция *TColorRef* преобразует значение цветовых составляющих в число типа *LongInt*.

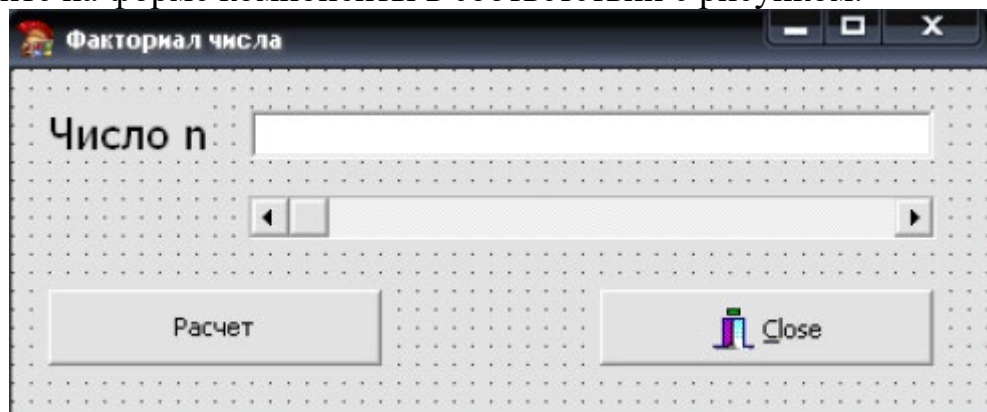
8. Аналогично запишите процедуры обработки события *OnChange* для элементов *GreenBar* и *BlueBar*.
9. Задайте имя формы проекта *RGB-функции*. Создайте кнопку *Close*.
10. Вызов процедур обработки событий *OnChange* поместите в событие при создании формы *OnCreate* для *Form1*.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  BlueBar.OnChange(Sender);
  RedBar.OnChange(Sender);
  GreenBar.OnChange(Sender);
end;
```

11. Сохраните изменения и запустите проект, убедитесь в его работоспособности.

Задание 2. Разработать проект, который позволяет пользователю вычислить факториал числа. Число, для которого рассчитывается факториал, выбирается с помощью вертикальной полосы прокрутки. При щелчке по кнопке «Расчет», меняется надпись «Число n» на «N!» и в строке ввода выводится значение факториала числа.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *ScrollBar2*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместите на форме компоненты в соответствии с рисунком.



4. Для элемента управления *ScrollBar1* установите значение свойств *Max=10*, *Position=5*.
5. Выделите элемент *ScrollBar1*, в *Object Inspector* перейдите на вкладку *Events*. Найдите событие *OnChange*, справа от него в поле сделать двойной щелчок левой кнопкой мыши. Оказавшись в коде программы, ввести следующий код:

```
procedure TForm1.ScrollBar1Change(Sender: TObject);  
begin  
  Edit1.Text:=IntToStr(ScrollBar1.Position);  
end;
```

6. Запишите процедуру обработки события *OnClick* кнопки «Расчет». При расчете факториала числа воспользоваться конструкцией цикла.
7. Задайте имя формы проекта *Факториал числа*. Создайте кнопку *Close*.
8. Сохраните изменения в проекте.

ПРАКТИЧЕСКАЯ РАБОТА № 20

Тема: Создание проекта с использованием группы зависимых переключателей

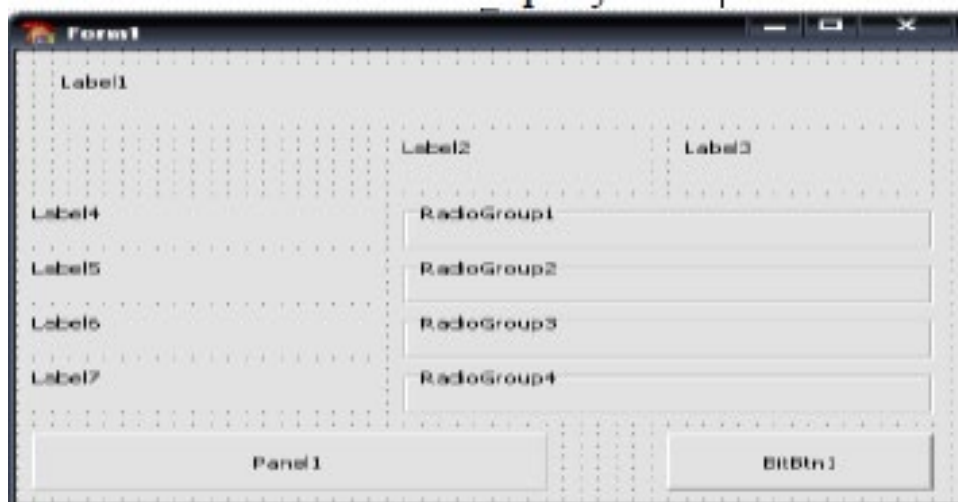
Цель работы: сформировать умения использования переключателей при создании проекта Delphi, изучить их основные свойства; сформировать умения по созданию процедур на основе событий компонентов.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать программу, с помощью которой пользователь мог бы выполнить следующее. После запуска программы появляется изображение, аналогичное рисунку. Пользователь по своему усмотрению выбирает один переключатель в группе. Каждому переключателю соответствует определенный балл. В зависимости от суммы набранных баллов появляется одно из сообщений «Вы пессимист», «Вы реалист» или «Вы оптимист».

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Radio1*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместить на форме компоненты в соответствии с рисунком.



4. Задайте для элементов управления *Label* значение свойства *Caption* в соответствии с рисунком. Для этого выделите элемент управления и перейдите в *Object Inspector* на страницу *Properties* (свойства).
5. Оставьте свойство *Caption* панели *Panel1* пустым.

6. Выделите компонент *RadioGroup1*, перейдите в *Object Inspector* на страницу *Properties*, найдите свойство *Caption* и удалите заголовок. Свойству *Columns*, определяющему количество колонок, в которые будут отображаться переключатели, присвойте значение 5.

7. Вызовите *String List Editor*, дважды щелкнув мышкой рядом со свойством *Items*, введите 5 строк со значениями 1, 2, 3, 4 и 5 соответственно.

8. Аналогичные действия проделайте с остальными компонентами *RadioGroup*. Можно выделить элементы управления с нажатой клавишей *Shift* и задать свойства, которые будут заданы для всех выделенных элементов управления.

9. Для того чтобы суммировать набираемые пользователем баллы, в процедуру обработки события *RadioGroup1.OnClick* вставьте следующий код:

```
sum:=0;
with RadioGroup1 do
  if ItemIndex>=0 then sum:=sum+ItemIndex+1;
```

Единицу необходимо прибавлять, т.к. индекс первого переключателя равен 0, но соответствует 1 баллу. Целочисленную переменную **sum** необходимо описать в разделе *var* модуля.

10. Вставьте в процедуру обработки событий *RadioGroup2.OnClick*, *RadioGroup3.OnClick* и *RadioGroup4.OnClick* аналогичные коды, но без обнуления переменной *sum*, т.к. обнуление необходимо лишь один раз перед началом суммирования.

11. Выведем на контрольную панель итоговое сообщение в зависимости от набранной суммы баллов. Для этого в процедуру обработки события *RadioGroup4.OnClick* добавим код:

```

case sum of
4..9:   Panel1.Caption:='Вы пессимист';
10..15: Panel1.Caption:='Вы реалист';
16..20: Panel1.Caption:='Вы оптимист';
end;

```

12. Выведем сообщение об окончании тестирования, добавив в процедуру обработки события *RadioGroup4.OnClick* код: *ShowMessage('Конец теста');*
В результате выполнения этого оператора появится информационное окно со словами «Конец теста» и единственной кнопкой *Ok*. После нажатия кнопки *Ok* в информационном окне сделайте недоступными все *RadioGroup*, а все переключатели в них - невыбранными. Для этого свойствам *Enabled* всех *RadioGroup* присвоить значение *False*, а свойствам *ItemIndex* - значение *-1*.
13. Для контроля правильности работы программы выведите на панель набранную пользователем сумму баллов. Для этого заголовку соответствующей панели присвоить значение *IntToStr(sum)*.
14. Запустите программу и убедитесь, что верная сумма баллов получается лишь при последовательном выборе переключателей сначала из *RadioGroup1*, затем из *RadioGroup2* и т.д. Если порядок был нарушен, или пользователь, изменив решение, выбрал другой переключатель в одной и той же группе, то результат будет неверным. Чтобы этого не случилось, сделайте доступной только ту группу переключателей, в которой необходимо сделать выбор. Для этого первоначально свойству *Enabled* всех *RadioGroup*, кроме *RadioGroup1*, присвойте значение *False*. В процедуре обработки события *RadioGroup1Click* добавить код, присваивающий свойству *Enabled RadioGroup2* значение *True* и т.д.
15. Сделайте возможным повторный запуск программы. Для этого разместите на форме кнопку *Button1*, свойству *Caption* которой присвойте значение *Retry*, а в процедуре обработки события *Button1Click* вставьте код, делающий доступной *RadioGroup1*.
16. Озаглавьте окно проекта *Проверь себя*.
17. Сохраните изменения и запустите проект, убедитесь в его работоспособности.

Задание 2. Разработать приложение, с помощью которого можно вычислить время падения тела с некоторой высоты при условии, что высота может задаваться в метрах, сантиметрах и дюймах.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Radio2*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместить на форме компоненты в соответствии с рисунком.

4. Создадим процедуру обработки события *Click* для кнопки *Вычислить*:

```
procedure TForm1.Button1Click(Sender: TObject);
  Const g=9.81;
  Var h,t : Real;
Begin
  h:=StrToFloat(Edit1.Text);
  Case radioGroup1.ItemIndex of
    0: t:=sqrt(2*h/g);           {Высота задается в метрах}
    1: t:=sqrt(2*h/100/g);      {Высота задается в санти-метрах}
    2: t:=sqrt(2*h*2.54/100/g); {Высота задается в дюймах}
  End;
  Edit2.Text:=FloatToStr(t);
end;
```

5. Запустите программу и проверьте её работоспособность.

6. Самостоятельно напишите обработчик события кнопки *Заккрыть*.

7. Недостаток компонента *RadioGroup* заключается в том, что имеется возможность определить только номер выбранной альтернативы, а не ее текстовое содержание. Для того чтобы определить и текстовое содержание альтернативы, можно использовать комбинированную строку, компонент *ComboBox*. Комбинированная строка ввода объединяет в себе свойство строки и списка. В обычном состоянии она имеет вид строки *Edit* со стоящей рядом

кнопкой с изображением направленной вниз стрелки. Если нажать эту кнопку, то появится список строк, где можно выбрать произвольную. Данный компонент имеет свойство *Items*, поэтому задание альтернатив, которые в данном случае будут раскрывающимся списком, происходит аналогичным образом. Однако на этапе выполнения допустимо свойство *Text*, в котором находится выбранная из списка строка. При работе с данным компонентом необходимо различать свойства *ComboBox.Items*, *Text* и *ComboBox.Text*. Если первое — это свойство, где находятся все строки списка, включая разделители (это свойство имеет подобный смысл и для *RadioGroup*), то второе содержит выбранную из списка строку.

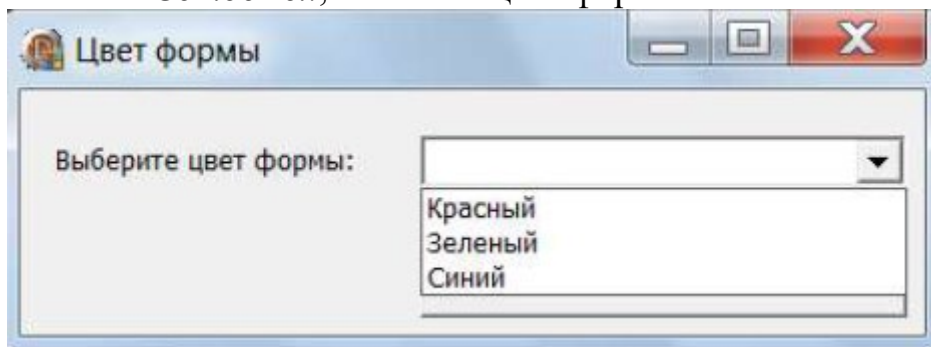
8. Измените проект так, чтобы выбор единицы измерения осуществлялся с помощью компонента *ComboBox*.

```
procedure TForm1.Button1Click(Sender: TObject);
  Const g=9.81;
  Var h,t : Real;
Begin
  h:=StrToFloat(Edit1.Text);
  if ComboBox1.Text='метры' then t:=sqrt(2*h/g);
  {Высота задается в метрах}
  if ComboBox1.Text='сантиметры' then t:=sqrt(2*h/100/g);
  {Высота задается в сантиметрах}
  if ComboBox1.Text='дюймы' then t:=sqrt(2*h*2.54/100/g);
  {Высота задается в дюймах}
  Edit2.Text:=FloatToStr(t);
end;
```

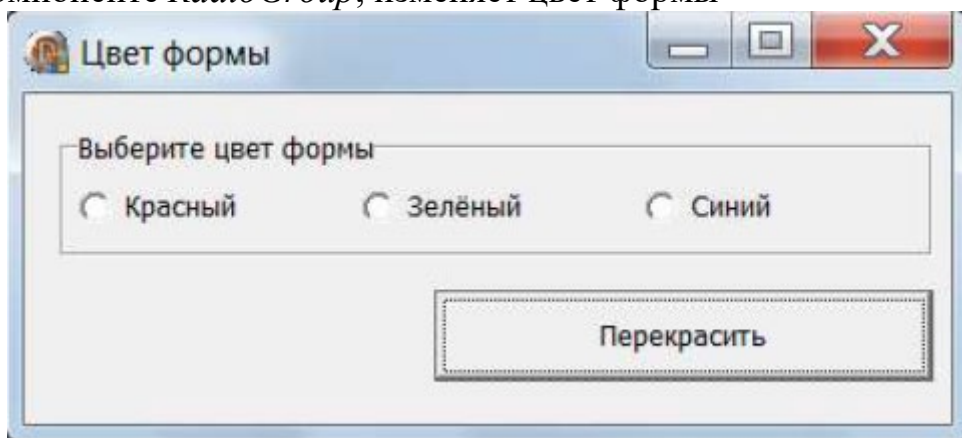
9. Выбор между компонентами *RadioGroup* и *ComboBox* во многом зависит от вкусов программистов, поскольку они выполняют одинаковую роль, а имеют различные внешний вид и применение. Необходимо отметить, что в языке *Delphi* есть большое количество компонентов, имеющих одинаковую область применения, однако отличающихся некоторыми частными внешними особенностями и доступными для применения методами.

10. Сохраните изменения в проекте.

Задание 3. Разработайте приложение, которое при выборе определенного цвета в компоненте *ComboBox*, изменяет цвет формы.



Задание 4. Разработайте приложение, которое при выборе определенного цвета в компоненте *RadioGroup*, изменяет цвет формы



ПРАКТИЧЕСКАЯ РАБОТА № 21

Тема: Создание процедур на основе событий

Цель работы: сформировать умения по созданию процедур на основе событий компонентов.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

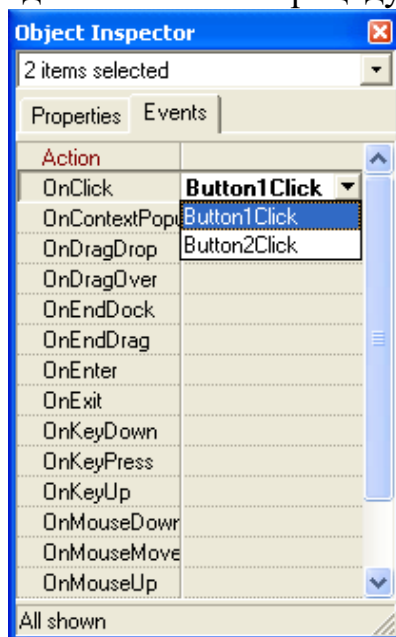
Справочный материал:

Событие (*Event*) – это то, что происходит во время работы программы. В Delphi каждому событию присвоено имя. Например, щелчок кнопкой мыши – это событие *OnClick*, двойной щелчок мышью событие *OnDblClick*.

Реакцией на событие должно быть какое-либо действие. В Delphi реакция на событие реализуется как **процедура обработки события**. Таким образом, для того чтобы программа выполняла некоторую работу в ответ на действия пользователя, программист должен написать процедуру обработки соответствующего события. Следует обратить внимание на то, что значительную часть обработки событий берет на себя компонент. Поэтому программист должен разрабатывать процедуру обработки события только в том случае, если реакция на событие отличается от стандартной или не определена.

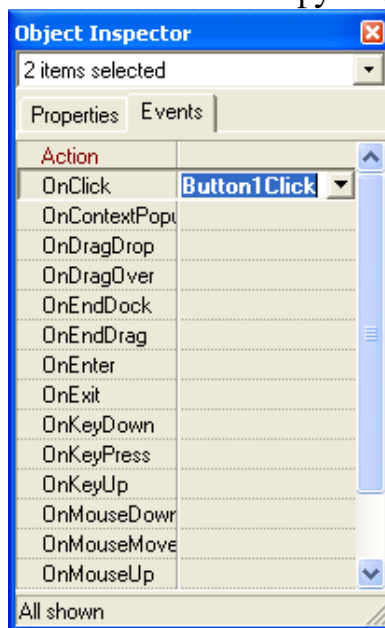
Чтобы приступить к созданию процедуры обработки события, надо сначала в окне *Object Inspector* выбрать компонент, для которого создается процедура обработки события. Затем в этом же окне нужно выбрать вкладку *Events* (События).

В левой колонке вкладки *Events* перечислены имена событий, которые может воспринимать выбранный компонент (объект). Если для события определена (написана) процедура обработки события, то в правой колонке рядом с именем события выводится имя этой процедуры.



Для того чтобы создать функцию обработки события, нужно сделать двойной щелчок мышью в поле имени процедуры обработки соответствующего события. В результате этого откроется окно редактора кода, в которое будет

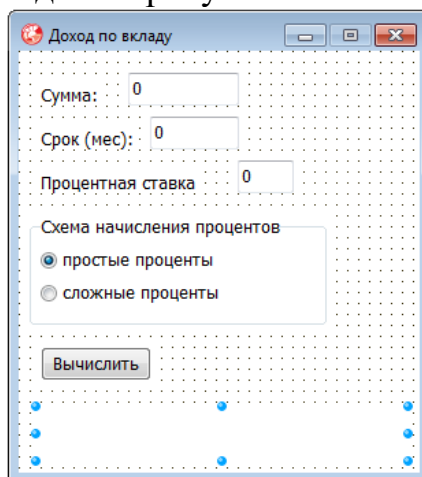
добавлен шаблон процедуры обработки события, а в окне *Object Inspector* рядом с именем события появится имя функции его обработки



Delphi присваивает функции обработки события имя, которое состоит из двух частей. Первая часть имени идентифицирует форму, содержащую объект (компонент), для которого создана процедура обработки события. Вторая часть имени идентифицирует сам объект и событие.

Содержание работы:

Задание 1. Напишите программу, которая вычисляет доход по вкладу. Программа должна обеспечивать расчет простых и сложных процентов. Простые проценты начисляются в конце срока вклада, сложные — ежемесячно и прибавляются к текущей (накопленной) сумме вклада и в следующем месяце проценты начисляются на новую сумму. Рекомендуемый вид формы программы приведен на рисунке:



1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Event1*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместите на форме компоненты в соответствии с рисунком.
4. Создайте процедуру обработки события *Click* для кнопки *Вычислить*:

// щелчок на кнопке Вычислить

procedure TForm1.Button1Click(Sender: TObject);

var

sum : real; *// сумма вклада*

pr: real; *// процентная ставка*

period: integer; *// срок вклада*

profit: real; *// доход по вкладу*

sum2: real; *// сумма, при вычислении методом сложных процентов*

i: integer;

begin

// получить исходные данные sum:= StrToFloat(Edit1.Text);

pr:= StrToFloat(Edit2.Text); period:= StrToInt(Edit3.Text);

if RadioButton1.Checked **then** profit := sum * (pr/100/12) *period

// выбран переключатель Простые проценты

else

*// т. к. в группе два переключателя, то если не выбран RadioButton1, то
 выбран*

// RadioButton2 — Сложные проценты

begin

 sum2:= sum;

for i:=1 **to** period **do** sum2 := sum2 + sum2 *(pr/100/12);

// здесь sum2 — сумма в конце срока вклада

 profit := sum2 — sum;

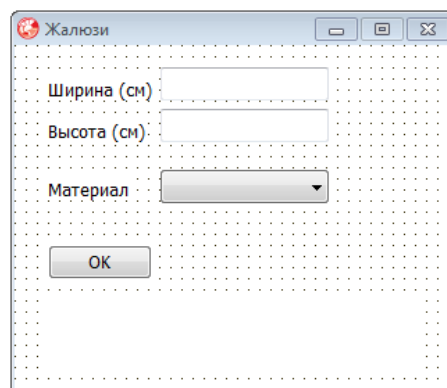
end;

 sum := sum + profit; Label4.Caption := 'Доход: ' FloatToStrF(profit,ffCurrency,6,2)
 +#13 +'Сумма в конце срока вклада: ' + FloatToStrF(sum,ffCurrency,6,2);

end;

5. Запустите программу и проверьте её работоспособность.

Задание 2. Написать программу, которая вычисляет стоимость жалюзи в зависимости от размера и материала (пластик, алюминий, текстиль, бамбук, соломка), из которого они изготовлены. Рекомендуемый вид формы приведен на рисунке.



ПРАКТИЧЕСКАЯ РАБОТА № 22

Тема: Создание процедур на основе событий

Цель работы: сформировать умения по созданию процедур на основе событий компонентов.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Написать программу, которая вычисляет стоимость автомобиля с учетом дополнительных опций. Рекомендуемый вид формы приведен на рисунке

Задание 2. Напишите программу, которая позволяет рассчитать тариф ОСАГО (обязательное страхование гражданской ответственности владельца транспортного средства). Рекомендуемый вид формы приведен на рисунке

ПРАКТИЧЕСКАЯ РАБОТА № 23

Тема: Создание проекта с использованием кнопочных компонентов

Цель работы: сформировать навыки разработки приложений с использованием кнопочных компонентов в среде визуального программирования Delphi, изучить особенности их использования

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Используя кнопочные компоненты *TSpeedButton*, разработать программу – калькулятор, выполняющий простейшие действия.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Калькулятор*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...* под именем *Calc.bdsproj*.
3. Задайте для формы следующие свойства: *Caption* – *Calc*, *Font - Size* – 10.
4. Разместите на форме одно поле ввода, дайте ему имя *Disp* и очистите свойство *Text*.
5. Для калькулятора мы будем использовать кнопки *TSpeedButton* (группа *Additional*). Их отличие от кнопок *TButton* и *TBitBtn* состоит в том, что они не получают фокуса ввода клавиатуры. Это значит, что их нельзя сделать активными, переходя от элемента к элементу с помощью клавиши **Tab**, и вообще их можно привести в действие только мышкой. Таким образом, на форме будет всего один компонент, принимающий сигналы с клавиатуры — это поле ввода.
6. Добавьте на форму 12 кнопок *TSpeedButton* для цифр, арифметических действий, знака «равно» и операции «сброс». Установите для всех кнопок размеры 25 на 25 пикселей и разместите их так, как на рисунке.



7. Дайте кнопкам-действиям имена *btnPlus*, *btnMinus*, *btnMul*, *btnDiv* (сложение, вычитание, умножение и деление), а кнопке «равно» — имя *btnCalc*.
8. Для действий и кнопки *C* установите жирный шрифт, а для кнопки *C* дополнительно — красный цвет шрифта.

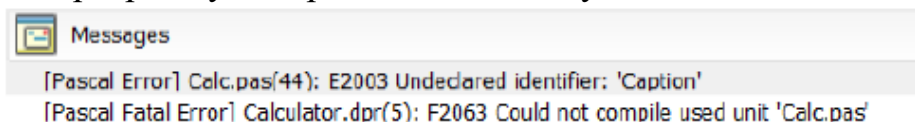
9. Кнопка *C* должна просто стирать содержимое поля ввода *Disp*. То есть, нужно вызвать метод *Disp.Clear*. Добавьте обработчик события *OnClick* для кнопки *C*.

10. Понятно, что когда пользователь щелкнул по кнопке-цифре, нужно добавить эту цифру в конец текста поля ввода. Конечно, можно для каждой кнопки написать обработчик типа *Disp.Text := Disp.Text + '1'*;

Однако можно сделать и более грамотно, определив для всех кнопок общий обработчик. Заметим, что цифра, которую нужно добавить, это и есть текст кнопки, ее свойство *Caption*. Таким образом, когда мы назначим всем кнопкам один единственный обработчик, в нем нужно как-то определить, какую именно кнопку мы нажали. А такая возможность есть, ведь обработчик имеет параметр *Sender* — это адрес компонента, от которого пришло сообщение. Выделите все кнопки-цифры и определите для них

общий обработчик события *onClick: Disp.Text := Disp.Text + Sender.Caption;*

11. Запустите программу. Скорее всего, вы получите сообщение об ошибке:

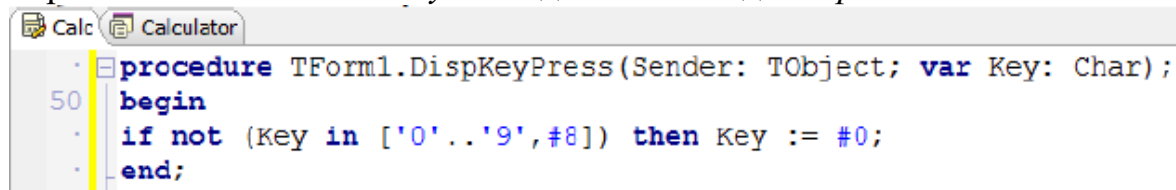


Фраза *Undeclared identifier* означает *Необъявленное имя*, то есть, по мнению *Delphi* объект *Sender* не имеет свойства *Caption*. Обратим внимание на то, что параметр *Sender* объявлен в заголовке процедуры как переменная типа *TObject*, а объект *TObject* действительно не имеет свойства *Caption* (это можно проверить по справочной системе). Но мы знаем, что в нашей программе все компоненты, которые могут вызвать этот обработчик, относятся к типу *TSpeedButton*, у которого свойство *Caption* есть.

Поэтому можно сказать, что программа должна воспринимать параметр **Sender** не как просто объект, а как *TSpeedButton*: *Disp.Text := Disp.Text + TSpeedButton(Sender).Caption;*

12. Исправьте код и запустите программу. Проверьте, как она работает при вводе текста и букв с клавиатуры и с помощью мыши.

13. При работе программы вы увидели, что с клавиатуры можно ввести буквы, которые нам совсем не нужны. Когда пользователь нажмет клавишу в поле ввода, возникает событие *OnKeyPress*, которое можно перехватить, установив соответствующий обработчик. В нем все «ненужные» символы заменяются символом с кодом 0, который не изменяет содержимое поля. Создайте обработчик события *OnKeyPress* для поля ввода *Disp*:



Разберемся, как работает данная процедура. Обратите внимание, что второй параметр обработчика объявлен как *var Key*, то есть, его можно менять в процедуре. В квадратных скобках записывают множество значений, причем '0'..'9' означает интервал: все символы от '0' до '9'. Символ с кодом 8,

обозначаемый как #8 — это возврат каретки. Если *Key* — не цифра и не возврат каретки, в эту переменную записывается 0 — нажатие клавиши игнорируется.

Запустите программу и попробуйте вводить буквы.

14. При работе программы вы могли заметить, что при щелчке по кнопке курсор сразу перемещается в начало числа, а хотелось бы, чтобы он оставался справа. Мы сделаем так, чтобы при вводе очередной цифры с помощью кнопки курсор автоматически переводился в конец числа. У компонента *TEdit* есть свойства *SelStart* и *SelLength*, которые определяют начало и длину выделенной части (она обычно обозначается синим фоном). Если *SelLength=0* (ничего не выделено), тогда *SelStart* указывает на позицию курсора в поле ввода. Если установить это свойство равным длине текста, курсор встанет сразу за последним символом.

15. Добавьте в конец обработчика события *OnClick* код: *Disp.SelStart := Length(Disp.Text)*; Проверьте работу программы.

16. Далее необходимо организовать вычисления. Нам нужны две переменных для хранения чисел и одна символьная переменная, в которую будем записывать тип операции. Т.к. при расчетах могут получиться числа с дробной частью, для хранения чисел будем использовать вещественные переменные. В простейшем случае для хранения операции можно использовать переменную типа *Char*, но мы объявим ее как символьную строку, так как при доработке программы могут понадобиться и многосимвольные названия операций. Объявите в начале программы две вещественных переменные *x1* и *x2* типа *Double* и одну символьную строку *oper*.

17. Когда нажимаем на одну из кнопок-операций, нужно запомнить введенное число в переменной *x1* и тип операции в переменной *oper*. Тип операции легко узнать, обратившись к свойству *Caption*. Поэтому можно установить для всех кнопок-операций один обработчик. Выделите все кнопки-операции и создайте для них один обработчик события *OnClick*:

```
x1 := StrToFloat(Disp.Text);  
oper := TSpeedButton(Sender).Caption;
```

18. При нажатии на кнопку «Равно» нужно прочитать из поля ввода второе число и выполнить операцию. При этом первое число и тип операции уже должны находиться в переменных *x1* и *oper*. Т.к. в результате деления может получиться число с дробной частью, переменная для хранения результата (назовем ее *res*) тоже должна быть вещественной.

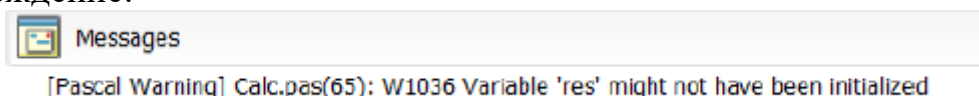
Введите обработчик события *OnClick* для кнопки «Равно»:

```
procedure TForm1.btnCalcClick(Sender: TObject);  
var res: double;  
begin  
  x2 := StrToFloat(Disp.Text);  
  if oper = '+' then res := x1 + x2;  
  if oper = '-' then res := x1 - x2;  
  if oper = '*' then res := x1 * x2;  
  if oper = '/' then res := x1 / x2;  
  Disp.Text := FloatToStr(res);  
end;
```

19. Запустите программу и проверьте ее работу. Учтите, что для ввода второго числа нужно сначала очистить экран кнопкой «С».

20. Закройте окно программы и затем запустите ее заново. Введите какое-нибудь число, а затем сразу щелкните по кнопке «равно». Что получилось? Попробуйте объяснить этот эффект, учитывая, что при создании глобальной символьной переменной *oper* в нее записана пустая строка.

21. Наверняка вы догадались, что проблема вызвана тем, что мы еще не задали операцию, а уже попытались что-то вычислить. При этом ни один условный оператор не сработал, и значение переменной *res* не изменялось. Переменная *res* объявлена в процедуре, то есть, она локальная. Память для локальных переменных выделяется в стеке при каждом новом вызове процедуры, причем эти ячейки не обнуляются. Поэтому в переменной *res* в нашем последнем эксперименте остался «мусор» — постороннее значение, которое программа и вывела на экран. Если вы были очень внимательны, можно было заметить, что при трансляции программы в окне *Message* (в нижнем левом углу) было выдано предупреждение:



Это значит «Переменной *res*, возможно, не будет присвоено никакого значения». Формально это не ошибка, и программа может запуститься, однако к предупреждениям нужно относиться внимательно, потому что они могут указать на скрытые логические ошибки (ошибки в алгоритме), как в нашем случае. Какой же выход? Проще всего сделать так: если в переменной *oper* записана пустая строка, мы просто выйдем из процедуры с помощью оператора *Exit*.

22. Добавьте в начало обработчика события *OnClick* для кнопки «Равно» строку:

```
if oper = '' then Exit;
```

Проверьте работу программы. Что вам не нравится?

23. Конечно, очень неудобно, что перед вводом второго числа нужно очищать поле ввода, нажимая на кнопку «С». Хотелось бы делать это автоматически.

24. Запустите стандартную программу *Калькулятор* и посмотрите, в какой момент стирается первое число. Наверное, вы увидели, что число из поля ввода автоматически стирается, когда после нажатия на кнопки-действия или кнопку «Равно» пользователь набирает новое число. Мы введем логическую переменную *newNumber*, которой будем присваивать значение *True* в том случае, если нужно начинать вводить новое число. Объявите глобальную логическую переменную *newNumber*.

25. В конце обработчиков события *OnClick* для кнопок-действий и кнопки «равно» добавьте строку: *newNumber := True*;

26. Очистку экрана будем делать перед вводом нового числа. Дополните обработчики кнопок-цифр. В начало обработчика события *OnClick* для кнопок-цифр добавьте код:

```
if newNumber then begin
    Disp.Clear;
    newNumber := False;
end;
```


27. Учтем в нашей программе, что пользователь может набирать число на клавиатуре. Для этого нужно дополнить аналогичным кодом обработчик события *OnKeyPress* для поля ввода. Измените обработчик события *OnKeyPress* так, как показано ниже.

28. Запустите программу и проверьте ее. Можно ли ввести знаки сложения, вычитания, умножения и деления с клавиатуры? Выполняется ли действие при нажатии на клавишу *Enter*?

```
if not (Key in ['0'..'9',#8]) then Key := #0
else
  if newNumber then begin
    Disp.Clear;
    newNumber := False;
  end;
```

29. При работе программы вы увидели, что пока программа позволяет вводить с клавиатуры только цифры, но не коды действий. Чтобы исправить этот недостаток, мы дополним обработчик *OnKeyPress*.

30. Что нужно сделать, когда нажали символ + на клавиатуре? Проще всего имитировать щелчок по кнопке, то есть, вызвать процедуру-обработчик события *OnClick* для кнопок-действий. Предположим, что этот обработчик называется *btnDivClick*. Но этой функции нужно передать параметр — адрес объекта, пославшего сообщение. Этот объект — кнопка *btnPlus*, поэтому при нажатии клавиши + нужно выполнить команду *btnDivClick (btnPlus)*;

Обработка нажатия других клавиш строится аналогично. При нажатии клавиши *Enter*, имеющей код 13 (он обозначается в программе как #13), нужно вызвать обработчик события *OnClick* для кнопки «равно» (она должна называться *btnCalc*).

31. Измените код обработчика *OnKeyPress* следующим образом:

```
procedure TForm1.DispKeyPress(Sender: TObject; var Key: Char);
begin
  if not (Key in ['0'..'9',#8]) then begin
    case Key of
      '+': btnMinusClick(btnPlus);
      '-': btnMinusClick(btnMinus);
      '*': btnMinusClick(btnMul);
      '/': btnMinusClick(btnDiv);
      #13: btnCalcClick(btnCalc);
    end;
    Key := #0;
  end
  else
    if newNumber then begin
      Disp.Clear;
      newNumber := False;
    end;
  Disp.SelStart := Length(Disp.Text);
  newNumber := True;
end;
```

32. Запустите программу и проверьте ее работу. После этого закройте проект.

ПРАКТИЧЕСКАЯ РАБОТА № 24

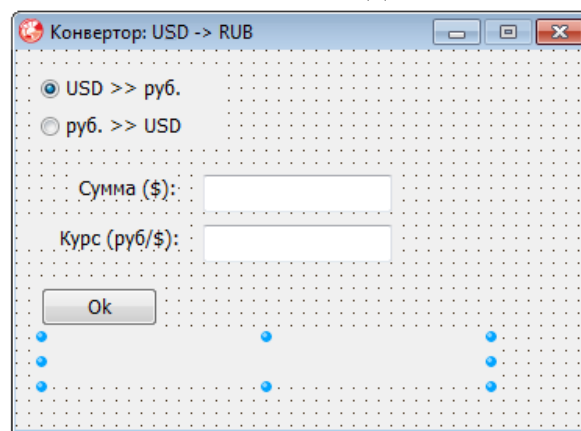
Тема: Создание проекта с использованием кнопочных компонентов

Цель работы: сформировать навыки разработки приложений с использованием кнопочных компонентов в среде визуального программирования Delphi, изучить особенности их использования

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Содержание работы:

Задание 1. Написать программу, которая позволяет пересчитать цену из долларов в рубли или из рублей в доллары. Рекомендуемый вид формы приведен на рисунке. Во время работы программы, в результате выбора вида конвертации, соответствующим образом должен меняться заголовок окна и текст, поясняющий назначение полей ввода.



Задание 2. Напишите программу *Таймер*. Форма программы приведена на рис. 1, а на рис. 2 — ее окно во время установки интервала и в процессе отсчета времени.

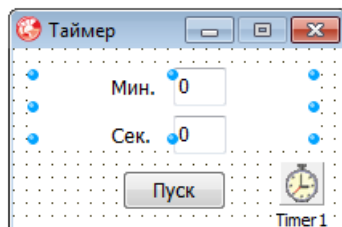
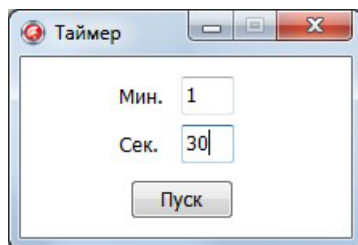
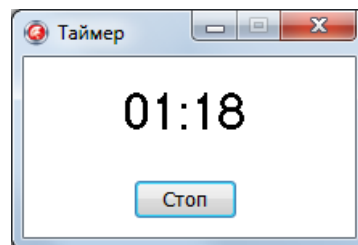


Рис. 1. Форма программы



а



б

Рис. 2. Окно программы *Таймер* во время установки интервала (а) и в процессе отсчета времени (б)

ПРАКТИЧЕСКАЯ РАБОТА № 25

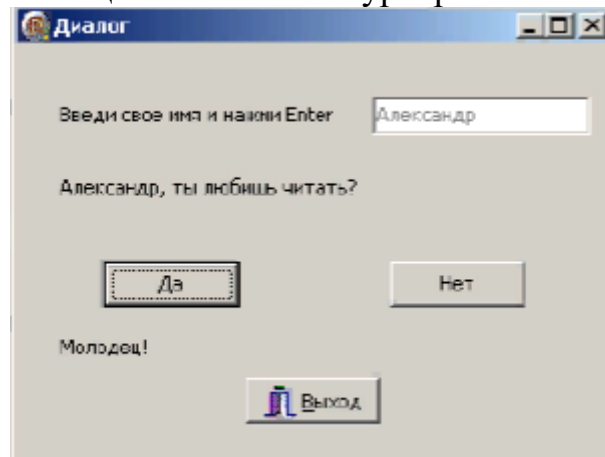
Тема: Создание проекта с использованием компонентов стандартных диалогов и системы меню.

Цель работы: сформировать умения использования компонентов стандартных диалогов и системы меню, изучить основные свойства; сформировать умения по созданию процедур на основе событий компонентов.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

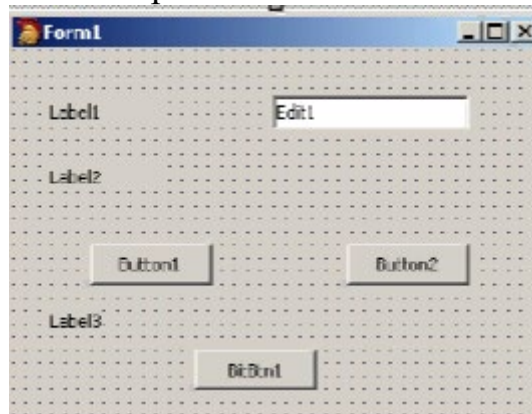
Содержание работы:

Задание 1. Создать программу, выполняющую следующие действия. После запуска программы пользователь вводит свое имя, например, Александр, в прямоугольник с мигающим текстовым курсором и нажимает клавишу *Enter*.



Появляется вопрос: «Александр, ты любишь читать?». Если пользователь щелкает на кнопке «Да», то появляется реплика «Молодец!», если на кнопке «Нет», то реплика «Почему же? Надо читать». Для выхода из программы необходимо щелкнуть мышью на кнопке «Выход».

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Диалог*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместите на форме экземпляры компонентов в соответствии с рисунке



4. Выполните следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ Имя события	Значение/Действие
Form1	Properties	Caption	Диалог
BitBtn1	Properties	Caption	&Выход
		Kind	bkClose
Label1	Properties	Caption	Введи свое имя и нажми Enter
Edit1	Properties	Caption	Удалить название объекта
	Events	OnKeyPress	If key=#13 then Label2.Caption:=Edit1.Text + ', ты любишь читать? ' ; Комментарий Каждая клавиша имеет свой код. Так клавиша Enter имеет код #13, из множества управляющих символов ASCII (American Standard Code For Information Interchange – американский стандартный код для обмена информацией).
Button1	Properties	Caption	Да
	Events	OnClick	Label3.Caption:= 'Молодец! ' ;
Button2	Properties	Caption	Нет
	Events	OnClick	Label3.Caption:= 'Почему же? Надо читать.' ;
Label2	Properties	Caption	Удалить название объекта
Label3	Properties	Caption	Удалить название объекта

5. Запустите программу и проверьте её работоспособность.

6. Сделайте кнопки «Да» и «Нет» доступными только после ввода имени и нажатия клавиши *Enter*.

Подсказка. Изменение свойства доступности компонента для пользователя – *Enabled*. Если свойство имеет значение *True*, то компонент доступен, а если значение *False*, то – не доступен. Значение свойства *Enabled* кнопок «Да» и «Нет» установите равными *False*, а в процедуру *Edit1KeyPress* включите, код:

Button1.Enabled := true;

Button2.Enabled := true;

7. Сделайте, чтобы строка ввода стала не доступной после нажатия кнопки «Да» или «Нет».

8. Введите дополнительную кнопку «Повторить», которая позволяет повторно выполнить задание.

Подсказка. Разместите на форме еще одну кнопку «Повторить». По нажатию кнопки «Повторить» программно очистите содержимое компонентов *Edit1*, *Label2*, *Label3* для повторного диалога:

Label2.Caption := '';

Label3.Caption := '';

Edit1.Text := '';

9.Сделайте так, чтобы при повторении диалога строка ввода была бы снова активной, используя *Form1.ActiveControl := Edit1*.

ПРАКТИЧЕСКАЯ РАБОТА № 26

Тема: Создание проекта с использованием компонентов стандартных диалогов и системы меню.

Цель работы: сформировать умения использования компонентов стандартных диалогов и системы меню, изучить основные свойства; сформировать умения по созданию процедур на основе событий компонентов.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

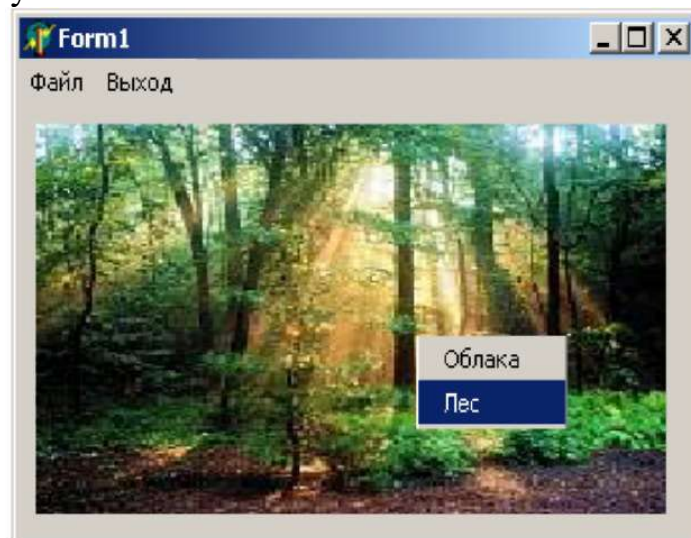
Содержание работы:

Задание 1. Создать программу, выполняющую следующие действия:

После запуска программы в окне изображается строка меню (*Файл, Выход*). При выборе пункта меню *Файл* появляются пункты меню (*Рисунки, Выход*). При выборе пункта меню *Рисунки* появляется вложенное меню, состоящее из двух пунктов (*Облака, Лес*).



По щелчку правой кнопки мыши появляется контекстное меню. Выбрать по пункту другой рисунок



Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка. Если выбрать любой из пунктов Выход, работа программы завершается.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.

2. Создайте в папке своей группы новую папку и назовите её *Меню*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместить на форме экземпляры компонентов: панель *Panel*, рисунок *Image*, диалоговое окно *OpenDialog*.
4. Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/имя события	Действие
Form1	Properties	Caption	Установка имени формы "Мое меню"
	Events	OnMouseDown	<pre>var p:TPoint; begin p.X :=X; p.Y :=Y; p := ClientToScreen (p); PopupMenu1.Popup (p.X, p.Y); end;</pre>
Запустить редактор меню (дважды щелкнуть на значке меню на форме)			
Form1.MainMenu1	Properties (в окне Object Inspector не выбран никакой объект)	Caption	Ввести текст пункта меню - Файл, и нажать Enter. Система присвоит ему имя N1
Между существующими и будущими пунктами меню можно переключаться с помощью щелчка мыши или курсорных клавиш.			
Form1.MainMenu1	Properties	Caption	Ввести текст пункта меню - Выход, и нажать Enter. Система присвоит ему имя N2.
	Events (щелкнуть на пункте Выход в строке меню)	N2Click	Close;
Щелкните на пункте Файл. Редактор меню создал еще одну заготовку под этим пунктом. Это заготовка для меню, которое откроется при выборе пункта Файл в работающей программе. Используя заготовки, создайте в этом меню два пункта: Рисунки (система присвоит ему имя N3) и Выход (N4). Выберите в редакторе меню пункт Рисунки и нажмите комбинацию клавиш Ctrl + Вправо.			
N4: TMenuItem	Events	OnClick	Выберем из раскрывающегося списка уже существующую процедуру-обработчик N2Click
Form1.MainMenu1	Properties	Caption	Ввести текст пункта меню -

			Облака, и нажать Enter. Система присвоит ему имя N5.
N5: TMenuItem	Events (выбрать в строке меню на форме пункт Облака)	OnClick	Image1.Picture.LoadFromFile ('C:\Windows\Облака.bmp');
Form1.MainMenu1	Properties	Caption	Ввести текст пункта меню - Лес, и нажать Enter. Система присвоит ему имя N6.
N6: TMenuItem	Events (выбрать в строке меню на форме пункт Лес)	OnClick	Image1.Picture.LoadFromFile ('C:\Windows\Лес.bmp');
Закройте окно редактора меню и убедитесь, что теперь строка меню появилась в основной форме программы.			
PopupMenu (Вкладка Standard)	Properties	Caption	Ввести текст пункта меню -Облака, и нажать Enter. Система присвоит ему имя N7.
		Caption	Ввести текст пункта меню - Лес, и нажать Enter. Система присвоит ему имя N8.
N7	Events	OnClick	Выберем из раскрывающегося списка уже существующую процедуру-обработчик N5Click
N8	Events	OnClick	Выберем из раскрывающегося списка уже существующую процедуру-обработчик N6Click
Image (Вкладка Additional)	Properties	Stretch	Присвоить значение True

5. Сохраните проект, запустите и протестируйте его.

Код подпрограммы:

```

procedure TForm1.N2Click (Sender: TObject);
begin
Close; end;
procedure TForm1.N5Click (Sender: TObject); begin
Image1.Picture.LoadFromFile ('C:\Windows\Облака.bmp'); end;
procedure TForm1.N6Click (Sender: TObject); begin
Image1.Picture.LoadFromFile ('C:\Windows\Лес.bmp'); end;

```

```
procedure TForm1.FormMouseDown  
(Sender: TObject; Button: TMouseButton; Shift: TShiftState; X, Y: Integer);  
var p:TPoint; begin  
p.X :=X; p.Y :=Y;  
p := ClientToScreen (p); PopupMenu1.Popup (p.X, p.Y); end;
```

ПРАКТИЧЕСКАЯ РАБОТА № 27

Тема: Разработка функциональной схемы работы приложения

Цель работы: изучить принцип разработки функциональной схемы работы приложения в среде Delphi.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы.

Справочный материал:

На втором этапе разработки приложения для каждого компонента, размещенного на форме, разработчик должен определить нужную реакцию на те или иные действия пользователя (например, нажатие кнопки или выбор переключателя).

Функциональная схема работы приложения определяется процедурами, которые выполняются при возникновении определенных событий, происходящих при взаимодействии пользователя с управляющими элементами формы. Реакция на события присуща каждой форме и не зависит от назначения приложения и его особенностей.

Каждый компонент имеет свой набор событий, на которые он может реагировать. Вместе с тем, существуют две категории стандартных событий, определенных для всех визуальных компонентов:

- события, определенные для всех без исключения визуальных компонентов
- события, характерные только для оконных визуальных компонентов.

Процедура, связанная с несколькими событиями для различных компонент, называется общим обработчиком и вызывается при возникновении любого из связанных с ней событий.

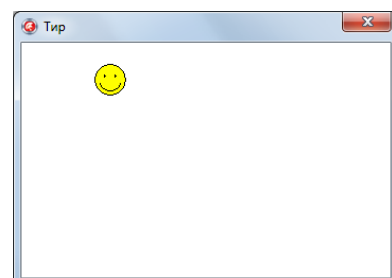
Для создания процедуры обработки события нужно:

- выделить на форме компонент;
- перейти на страницу событий Инспектора Объектов;
- выделить событие, для которого будет создаваться процедура-обработчик события;
- посредством двойного нажатия кнопки мыши в области значения события получить доступ в модуль формы, где Delphi автоматически создаст заготовку процедуры-обработчика;
- в месте, где будет установлен текстовый курсор, написать код, который должен выполняться при возникновении события.

Содержание работы:

Задание 1. Напишите программу *Тир*. В ее окне случайным образом должна "прыгать" цель, например веселая рожица, на которой пользователь может сделать щелчок кнопкой мыши. Движение рожицы должно прекратиться после того, как пользователь сделает 10 щелчков кнопкой мыши.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Тир*. Сохраните проект в созданную



ранее папку, выполнив команду *File - Save Project as...*

3. Напишите код программы:

type

 TForm1 = class(TForm)

 Timer: TTimer;

 Label1: TLabel;

 Button1: TButton;

procedure TimerTimer(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure FormMouseDown(Sender: TObject; Button: TMouseButton; Shift: ShiftState; X, Y: Integer);

procedure Button1Click(Sender: TObject);

private { Private declarations }

public { Public declarations }

{ объявление процедур помещено сюда, чтобы процедуры имели прямой доступ к форме, на которой они рисуют }

procedure PaintFace(x,y: integer); *// рисует рожицу*

procedure EraseFace(x,y: integer); *// стирает рожицу*

end;

var

 Form1: TForm1;

 fx,fy: integer; *// координаты рожицы*

 n: integer; *// количество щелчков кнопкой мыши*

 p: integer; *// количество попаданий*

implementation

// рисует рожицу

procedure TForm1.PaintFace(x,y: integer);

begin

 Canvas.Pen.Color := clBlack; *// цвет линий*

 Canvas.Brush.Color := clYellow; *// цвет закрашки*

// рисуем рожицу

 Canvas.Ellipse(x,y,x+30,y+30); *// лицо*

 Canvas.Ellipse(x+9,y+10,x+11,y+13); *// левый глаз*

 Canvas.Ellipse(x+19,y+10,x+21,y+13); *// правый глаз*

 Canvas.Arc(x+4,y+4,x+26,y+26,x,y+20,x+30,y+20); *// улыбка*

end; // стирает рожицу

procedure TForm1.EraseFace(x,y: integer);

begin

// зададим цвет границы и цвет закрашки, совпадающий с цветом формы.

 Canvas.Pen.Color := Form1.Color; *// цвет окружности*

 Canvas.Brush.Color := Form1.Color; *// цвет закрашки*

 Canvas.Ellipse(x,y,x+30,y+30);

end;

{ \$R *.dfm }

procedure TForm1.TimerTimer(Sender: TObject);

```

begin
EraseFace(fx,fy); // новое положение рожницы
fx:= Random(ClientWidth-30); // 30 — это диаметр рожницы
fy:= Random(ClientHeight-30);
PaintFace(fx,fy);
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
// исходное положение рожницы
fx:=100; fy:=100;
Randomize; // инициализация генератора случайных чисел
end;
// нажатие клавиши мыши
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
inc(n); // кол-во щелчков
if (x > fx) and (x < fx+30) and (y > fy) and (y < fy+30) then begin
// щелчок по рожнице
inc(p);
end;
Form1.Caption := 'Выстрелов: '+IntToStr(n) + ' Попаданий: ' + IntToStr(p);
if n = 10 then
begin
// игра закончена
Timer.Enabled := False; // остановить таймер
ShowMessage('Выстрелов: 10. Попаданий: ' + IntToStr(p)+'.');
EraseFace(fx,fy);
Label1.Visible := True; Button1.Visible := True;
// теперь кнопка и сообщение снова видны
end; end; // щелчок на кнопке Ok
procedure TForm1.Button1Click(Sender: TObject);
begin
n := 0; // выстрелов
p := 0; // попаданий
Label1.Visible := False; // скрыть сообщение
Button1.Visible := False; // скрыть кнопку
Timer.Enabled := True; // пуск таймера
end;

```

4. Сохраните проект и проверьте программу на работоспособность.

ПРАКТИЧЕСКАЯ РАБОТА № 28

Тема: Разработка функциональной схемы работы приложения

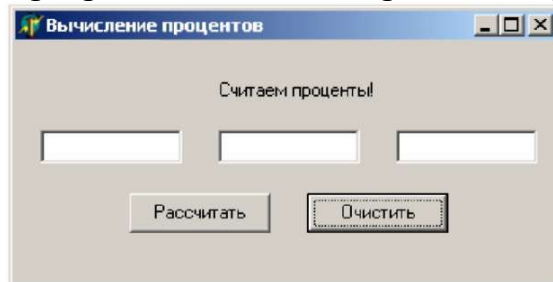
Цель работы: изучить принцип разработки функциональной схемы работы приложения в среде Delphi.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

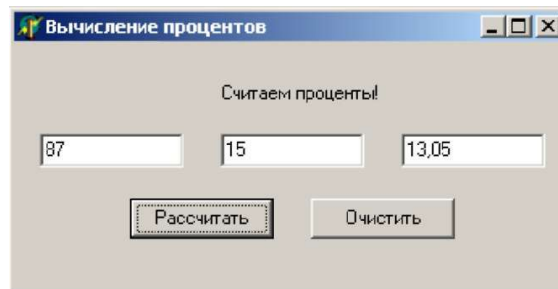
Содержание работы:

Задание 1. Создать программу, выполняющую следующие действия:

1. После запуска программы в окне изображается три текстовых поля.



2. В первое поле вводится число. Во второе поле - проценты. При нажатии кнопки "Рассчитать" в третье поле выводятся вычисленные проценты от числа.

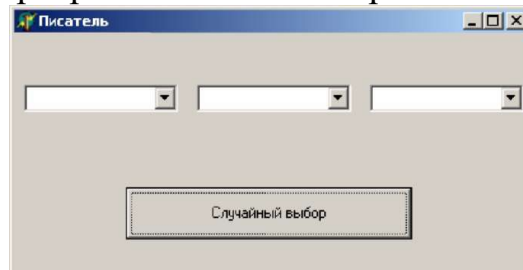


3. При нажатии кнопки "Очистить" очищаются значения полей. Далее вводятся новые значения в поля.

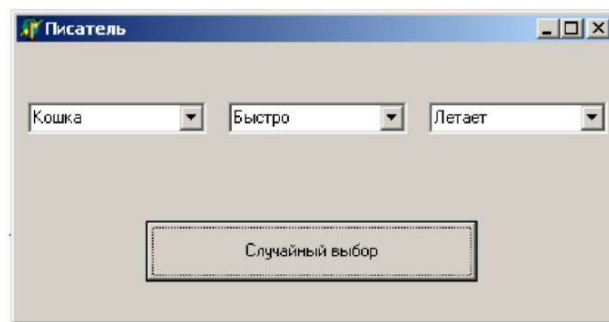
4. Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

Задание 2. Создать программу, выполняющую следующие действия:

1. После запуска программы в окне изображается три поля.



2. По щелчку мышью на кнопке "Случайный выбор" из трех слов составляется предложение случайным образом.



3. Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

ПРАКТИЧЕСКАЯ РАБОТА № 29

Тема: Разработка оконного приложения с несколькими формами

Цель работы: научиться создавать приложения, в которых используются несколько форм.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

Справочный материал:

Любая программа в Delphi имеет как минимум одну форму, которая называется главной. При формировании проекта автоматически создается одна форма. Если нет необходимости в создании нескольких форм, то в этом случае следует только задать главной форме необходимые характеристики и заполнить ее компонентами. Для добавления в проект новой формы следует воспользоваться командой меню *Flle – New Form*.

Все формы создаются на основе класса *TForm*. Класс *TForm* предоставляет возможность изменять поведение и внешний вид формы с помощью ряда свойств, методов и событий.

Таблица. Свойства класса TForm

Название	Описание
Active	Определяет состояние формы (True - активная)
BorderIcons	Отвечает за наличие кнопок в строке заголовка (biSystemMenu - значок вызова системного меню, biMinimize - кнопка минимизации, biMaximize - кнопка максимизации, biHelp - кнопка вызова справки, которая появляется на экране, если отключить biMinimize и biMaximize)
BorderStyle	Выбирает изображение рамки формы – bsNone - форма не имеет рамки, ее размер постоянен и отсутствуют кнопки системного меню, а также кнопки минимизации и максимизации окна; – bsSingle - размеры формы неизменны, имеется рамка толщиной 1 пиксель; – bsDialog - стандартная рамка, размеры нельзя изменять; – bsSizeable - размеры формы можно изменять, стандартная рамка; – bsToolwindow - сходный с bsSingle, но строка заголовка меньше; – bsSizeToolwin - сходный с bsSizeable, но заголовок по высоте уменьшен
Canvas	Обеспечивает доступ к форме для ее прорисовки
ClientHeight	Задаёт высоту формы
ClientWidth	Устанавливает ширину формы
HelpFile	Содержит имя файла помощи. По умолчанию - файл помощи приложения
Icon	Содержит пиктограмму, которая будет отображаться при минимизации формы
Menu	Содержит главное меню формы
ModalResult	Используется для закрытия модальных форм
Visible	Отвечает за видимость формы (True - видимая, False - невидимая)
WindowState	Отвечает за внешний вид формы на экране – wsNormal - обычные размеры формы; – wsMaximized - форма развернута на весь экран; – wsMinimized - форма свернута в значок на панели задач Windows

Таблица. Методы класса TForm

Название	Описание
Close	Закрывает форму (если она главная, то закрывает приложение)
CloseQuery	Определяет, можно закрыть сейчас форму или нет (True - можно)
FocusControl	Активизирует указанный компонент, помещенный в форму
Next	Делает активной следующую MDI-форму
Previous	Активизирует предыдущую MDI-форму
Print	Распечатывает форму на принтере
Release	Удаляет форму с экрана и из динамической области памяти
Show	Вызывает форму в немодальном режиме
ShowModal	Делает форму видимой и модальной

Таблица. События класса TForm

Название	Описание
OnActive	Возникает при активизации формы
OnClose	Вызывается перед закрытием формы. Параметр Action определяет последствия обработки события: caNone - форма не закрывается, caHide - спрятать форму, caFree - удаление из приложения и динамической памяти, caMinimize -минимизация формы
OnCloseQuery	Вызывается перед закрытием окна. Параметр CanClose определяет возможность закрытия окна
OnCreate	Появляется при создании формы, но перед ее появлением на экране
OnDeactivate	Генерируется, когда форма становится неактивной
OnDestroy	Обрабатывает событие, возникающее перед удалением формы из динамической памяти
OnHelp	Возникает при обращении к справке
OnHide	Срабатывает перед удалением формы с экрана
OnPaint	Является обработчиком события, возникающего при необходимости перерисовки формы
OnResize	Реагирует на изменение размеров формы
OnShow	Возникает, когда форма становится видимой на экране

Многостраничный блокнот PageControl.

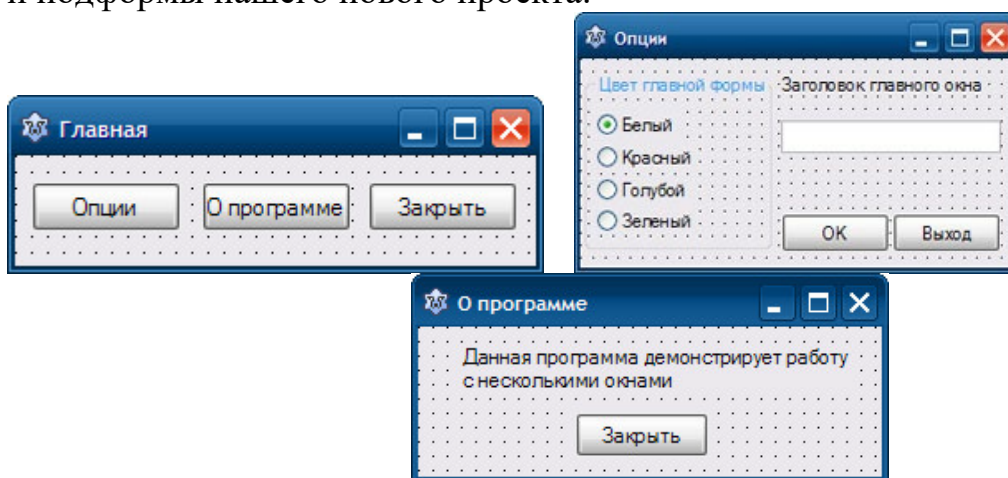
Компонент *PageControl* является управляющим элементом, включающим набор из нескольких страниц, размещаемых одна под другой. Каждая страница имеет закладку, которая является неотъемлемой частью данной страницы, в отличие от одностраничного блокнота. Страницы используются для объединения различных управляющих элементов в группы, обеспечивая их компактное размещение и простое переключение между ними. Многостраничный блокнот является более сложным управляющим элементом, чем компонент *TabControl*, однако многие свойства этих двух элементов совпадают. Поэтому будут рассмотрены только свойства, специфичные для компонента *PageControl*.

Свойство	Описание
ActivePage (тип TTabSheet)	определяет название текущей (выбранной) страницы компонента PageControl.
ActivePageIndex (тип Integer)	служит для определения индекса текущей страницы. С помощью свойств ActivePage и ActivePageIndex можно программно устанавливать новую активную страницу
PageCount (тип Integer)	позволяет определить количество страниц многостраничного блокнота.
Pages [Index: Integer] (тип TTabSheet)	представляет собой список всех страниц управляющего элемента PageControl. Используя данное свойство во время выполнения приложения, можно получить доступ к любой странице блокнота по ее номеру, задаваемому параметром index.

Добавление и удаление страниц, а также перемещение между страницами компонента *PageControl* в процессе проектирования приложения осуществляется с помощью вызова контекстного меню (нажатием правой кнопки мыши в поле компонента) и дальнейшего выбора соответствующего пункта меню. Перемещаться между страницами можно также простым нажатием на закладке необходимой страницы. Это возможно, потому что каждая страница (включая закладку) является отдельным независимым объектом.

Содержание работы:

Задание 1. Создать приложение с тремя формами: *Главная*, *Опции* и *О программе*. Форму *Опции* вызывать в обычном окне. Для вызова формы *О программе* использовать модальное окно. На рисунке показаны главная форма и подформы нашего нового проекта.



1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Три формы*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Разместите на форме *Form1* 3 кнопки, измените свойства объектов в соответствии с таблицей.

Компонент	Свойство	Значение	Форма
Name	Caption	frmGeneral	Главная
Кнопка 1	Caption	Опции	
Кнопка 2	Caption	О программе	
Кнопка 3	Caption	Закрыть	


4. Создайте форму *Form2*. Для этого выберите в меню *Файл* команду *Создать форму*. На экране появится новая форма *Form2*, в редакторе кода – новая вкладка *Unit2*.

5. Установите на форме компонент *RadioGroup* для выбора цвета, надпись, поле вывода, две кнопки. Настройте свойства объектов в соответствии с таблицей.

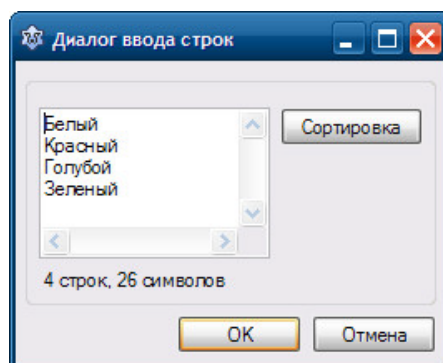
Компонент	Свойство	Значение	Форма
Name	Caption	frmOptions	Опции
RadioGroup	Caption	Items	Цвет главной

			формы
Label1	Caption		Заголовок главного окна Ввести список
Edit1	Text		Пусто
Button1	Caption		ОК
Button2	Caption		Выход

6. После размещения на форме компонента *TradioGroup*, входящие в него переключатели задаются перечислением их названий. Эти названия вводятся в свойство *Items*.

7. Так как требуется ввести не одну строку, а несколько, для их ввода предусмотрен специальный редактор, который вызывается щелчком на специальной кнопке , расположенной справа в строке, описывающей свойство *Items*.

8. На рисунке ниже показан редактор списка для ввода названий переключателей.



9. Большая текстовая область окна редактора предназначена для ввода названий переключателей по одному в каждой строке. Переход в начало следующей строки осуществляется при нажатии на клавиши *Shift+Enter*.

10. После окончания ввода списка, щелкните по кнопке *ОК*, и внешний вид объекта *RadioGroup1* на форме сразу изменится.

11. Создайте еще одну форму – *Form3*, выбрав команду *Файл - Создать форму*. На экране появиться новая форма *Form3* а в редакторе кода – новая вкладка *Unit3*.

12. Разместите на *Form3* объекты *Надпись* и *Кнопка*. Настройте свойства объектов.

Компонент	Свойство	Значение	Форма
Name	Caption	frmAbout	О программе
Button1	Caption		Выход

13. Программный код для формы *Главная* (модуль *Unit1*) В модуле *Unit1* в разделе *Implementation* необходимо записать директиву *uses*:

`uses Unit2, Unit3;`

Это необходимо для того чтобы модули *Unit2*, *Unit3* форм *Опции* и *О программе* были видимы в главном модуле *Unit1*.

14. Написать обработчики событий для кнопок формы *Главная*. Первая кнопка формы *Главная* (кнопка *Опции*) вызывает форму *Опции* в обычном окне с помощью метода *Show*.

```
procedure TfrmGeneral.Button1Click(Sender: TObject);  
begin  
    frmOptions.Show;  
end;
```

15. Вторая кнопка формы *Главная* (кнопка *О программе*) вызывает форму *О программе* в модальном окне с помощью метода *ShowModal*.

```
procedure TfrmGeneral.Button2Click(Sender: TObject);  
begin  
    frmAbout.ShowModal;  
end;
```

16. Третья кнопка формы *Главная* (Кнопка *OK*) закрывает главное окно.

```
procedure TfrmGeneral.Button3Click(Sender: TObject);  
begin  
    Close;  
end;
```

17. Открываем программный код формы *Опции* (модуль *Unit2*). В модуле *Unit2* в разделе *implementation* записать директиву *uses*:

```
uses Unit1;
```

Это необходимо для того чтобы главный модуль *Unit1* формы *Главная* был видим в этом модуле.

18. Создать обработчик загрузки формы *Опции*, в который записать программный код, передающий текст заголовка главной формы в поле *Edit1*.

```
procedure TFrmOptions.FormCreate(Sender: TObject);  
begin  
    frmOptions.Edit1.text:=frmGeneral.Caption;  
end;
```

19. Кнопка *OK* формы *Опции*. По щелчку на этой кнопке будет происходить изменение цвета главной формы.

```
procedure TFrmOptions.Button1Click(Sender: TObject);  
begin  
    if radioGroup1.ItemIndex=0 then frmGeneral.color:=clWhite;  
    if radioGroup1.ItemIndex=1 then frmGeneral.color:=clRed;  
    if radioGroup1.ItemIndex=2 then frmGeneral.color:=clBlue;  
    if radioGroup1.ItemIndex=3 then frmGeneral.color:=clGreen;  
end;
```

20. Кнопка *Заккрыть* формы *Опции*. По щелчку на этой кнопке закрывается окно *Опции*.

```
procedure TFrmOptions.Button2Click(Sender: TObject);  
begin  
    close;  
end;
```

21.Переходим в программный код формы *О программе* (модуль Unit3).В модуле *Unit3* в разделе *implementation* записать директиву *uses*.

```
uses Unit1;
```

Модуль *Unit1* формы *Главная* должен был видим в этом модуле.

22.Кнопка *OK* формы *О программе* закрывает окно.

```
procedure TfrmAbout.Button1Click(Sender: TObject);
```

```
begin
```

```
Close;
```

```
end;
```

23.Проект готов. Сохраните проект и проверьте его работу.

ПРАКТИЧЕСКАЯ РАБОТА № 30

Тема: Разработка оконного приложения с несколькими формами

Цель работы: научиться создавать приложения, в которых используются несколько форм.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

Содержание работы:

Задание 1. Разработать многооконное приложение.

1. На главную форму приложения поместить компонент *TAnimate*. Компонент должен реализовать просмотр пользовательской анимации, путь которому прописывается в свойстве *FileName*.

2. На главной форме реализовать прослушивание музыкальной композиции при помощи компонента *TMediaPlayer*.

3. Поместить на форму главное меню, содержащее следующие пункты:

- задание 1
- задание 2
- выход

При выборе пункта меню «Задание 1», необходимо реализовать открытие формы «График» (в модальном режиме). На форме «График» разместить 2 вкладки:

- «Таблица»- содержит таблицу, содержащее протабулированное значение функции $y=f(x)$ на отрезке $[a, b]$. Количество значений функций равно n . Шаг вычисляется по формуле $h=(b-a)/n$.
- «График» - содержит график функции $y=f(x)$, начерченный с помощью компонента *TChart*.

При выборе пункта меню «Задание 2», необходимо открытие формы «Теория» (в немодальном режиме). На форме «Теория» расположить 2 компонента *TreeView* и *WebBrowser*. Узлами дерева *TreeView* являются темы теоретического материала. По щелчку названию темы необходимо осуществить загрузку соответствующей страницы *html* в компонент *WebBrowser*.

При выборе пункта меню «Выход», необходимо закрыть приложение.

№	Функция	Интервал	n
1.	$Y=2\sin(x/3)$	$[-\pi/2; \pi/2]$	30
2.	$Y=2\cos(x/4)$	$[0; 3\pi/2]$	40
3.	$Y= \sin(x)+\cos(x) $	$[0; \pi]$	40
4.	$Y= \sin(x)-\cos(x) $	$[0; \pi]$	40
5.	$Y=2\sin(x)+3\cos(x)$	$[-\pi; \pi]$	50
6.	$Y=\sin(x)+\cos^2(x)$	$[-\pi; \pi]$	50
7.	$Y=2-\cos(x)$	$[0; 3\pi/2]$	40
8.	$Y=\sin(\sqrt{2}x)+\cos(x)$	$[0; 2\pi]$	50
9.	$Y=2\sin(2x)+1$	$[-\pi/2; \pi/2]$	50
10.	$Y=\sin(x)+\cos(x)-1$	$[-\pi; \pi]$	40

ПРАКТИЧЕСКАЯ РАБОТА № 31

Тема: Разработка игрового приложения

Цель работы: научиться разрабатывать игровые приложения.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

Содержание работы:

Задание 1. Создать программу – игру. Игрок управляет пушкой зенитки, его боевое задание – справиться с нашествием воздушных шаров. Воздушные шары несут бомбы, которые они сбросят, как только окажутся над пушкой. Необходимо не допустить этого и уничтожить их все на подлете. Снаряды не ограничены, но следующий выстрел можно делать только после того, как выпущенный снаряд поразит цель, упадет на землю или уйдет из зоны видимости.

В игре участвуют: воздушные шары, зенитная пушка, пушечный снаряд, бомба, внешняя среда.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Игра*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. Для шаров введен специальный тип *TBalloon*., в котором содержатся данные о координатах шара, его скорости, состоянии и цвете. Массив переменных *Balloons* типа *TBalloon* будет содержать полную информацию обо всех шарах. Индексирование идет, начиная с нуля. Общее число шаров в массиве задается соответствующей константой.

Кроме того, к шарам относятся константы, определяющие: количество шаров, их возможные цвета, возможную высоту над землей (всего предполагается четыре уровня), интервал между шарами и их радиус.

Type *TBalloon* = **record**

x, y, v, Explosion: integer;

Color: TColor;

end;

const BallCount = 10;

BallColors: **array** [0 .. 9] **of** TColor = (clRed, clGreen, clNavy, clMaroon, clPurple, clOlive, clLime, clYellow, clFuchsia, clSilver);

BallAltitude: **array** [0 .. 3] **of** integer = (240, 160, 200, 120);

BallInterval = 40; BallRadius = 15;

V = 150; g = -9.8 * 3; dt = 0.1;

var

Form1: TForm1;

x, y, Vx, Vy, BombY, BombV: double;

Angle, GunPosition, GunExplosion: integer;

Balloons: **array** [0 .. BallCount - 1] **of** *TBalloon*;

4. Зенитная пушка описывается переменными: угол наклона пушки, ее положение, ее состояние.

Пушечный снаряд описывается переменными: координатами, текущей скоростью, начальной скоростью при выстреле.

Бомба, сбрасываемая с воздушного шара, имеет: координату, скорость.

Для расчета положений объектов нужно знать ускорение падения и шаг по времени.

В качестве состояния шара и пушки мы используем переменные *TBalloon*. В определенный момент времени каждый элемент должен находиться в каком-то состоянии, определяющем то, как элемент выглядит на экране. Шары и пушка могут быть боеспособными, взрывающимися и уничтоженными.

Если *Explosion* = 0 - шар боеспособен, если *Explosion* = 10, то он уничтожен, а значения от 1 до 9 обозначают фазы взрыва. Чтобы инициировать взрыв боеспособного шара, нужно присвоить *Explosion* := 1. То же самое касается и пушки.

Состояние, в котором скорость снаряда равна нулю: $V_x = 0$ и $V_y = 0$. В этом случае считают, что можно сделать новый выстрел. Пока же снаряд летит (т.е. скорость его не нулевая), новый выстрел сделать нельзя.

Бомба будет сбрасываться шаром, когда он поравняется с пушкой. Таким образом, *X* - координата бомбы всегда равна *X* - координате пушки. Следовательно, необходимо хранить только *Y*-координату бомбы. Бомба не сброшена, если *BombY* = 1000. Если *BombY* < 1000, то она считается сброшенной и начинает падать.

5. Поместите на форму *TrackBar*, *Button*, *Image* и *Timer* и настройте их параметры.

Timer: Interval = 100; *Image*: Width = 440, Height = 260; *Button*: Caption = 'Новая игра'; *TrackBar*: Min = -90, Max = 90, Frequency = 10.

Image - сражение в реальном времени. *Timer* - реальное время. *Button* - запуск новой игры. *TrackBar* - управление зенитной пушкой.

В обработчике нажатия на кнопку "Новая игра" необходимо обнулить параметры, расставить пушку и шары. Для каждого шара устанавливается случайное направление движения: $v = 1$ или -1 . В зависимости от направления, определяется высота, на которой находится шар (на одном и том же уровне шары летят в одну сторону). По *x*-координате шары расставляются так, чтобы они шли с примерно равным интервалом в направлении пушки из-за границ экрана.

Каждый шар *Balloons[i]*, элемент массива *Balloons*, имеет тип записи *TBalloon*, и для того, чтобы получить доступ к его элементам, нужно писать: *Balloons[i].Explosion* := 0, *Color* := *BallColors[random(10)]* и т.д. Конструкция **with** *Balloons[i]* **do begin .. end** позволяет избежать повторения.

```
procedure TForm1.Button1Click(Sender: TObject);  
var i: integer;  
begin  
  Caption := 'Нашествие';  
  TrackBar1.Position := 0;  
  GunPosition := 170 + random(100);  
   $V_x := 0$ ;  $V_y := 0$ ;
```

```

GunExplosion := 0;
for i := 0 to BallCount - 1 do
with Balloons[i] do
begin
if random < 0.5 then v := -1 else v := 1;
y := BallAltitude[1 + v + random(2)] + random(10);
x := 220 - v * (220 + i * BallInterval + random(20));
Explosion := 0;
Color := BallColors[random(10)];
end;
BombY := 1000;
end;

```

6. В обработчик события формы *OnCreate* вводится датчик случайных чисел, устанавливается клиентский размер формы (*ClientWidth* := 455; *ClientHeight* := 315;). Размер клиентской области формы в приложениях определяется автоматически как *Width u Height* формы минус ширина заголовка и бордюров.

```

procedure TForm1.FormCreate(Sender: TObject);
begin
Randomize;
ClientWidth := 455; ClientHeight := 315; Button1.Click;
end;

```

7. Пушка танка будет управляться с помощью компонента *TrackBar*. Выстрел производится нажатием пробела на клавиатуре. При этом фокус ввода должен быть у *TrackBar*. В обработчике *OnKeyPress* реализуется выстрел, задавая начальную скорость и положение снаряду, а также издавая звук выстрела.

Процедура, проигрывающая этот звук из wav-файла, описана в модуле, который необходимо подключить в разделе *uses*. В модуле *MMSystem* имеется функция *PlaySound*, с помощью которой можно проигрывать файлы *wav*. Для асинхронного воспроизведения (приложение не приостанавливает работу на время воспроизведения, а проигрывает его в фоновом режиме) вызов выглядит так: *PlaySound(PChar('имяфайла.wav'), 0, SND_ASYNC)*.

Найдите звуки для выстрела (условно назовем *GunShot.wav*) и взрыва (условно назовем *Explode.wav*) и скопируйте их в папку, в которой сохранен проект.

```

procedure TForm1.TrackBar1KeyPress(Sender: TObject; var Key: Char);
var i: integer;
begin
if (Key = ' ') and (Vx = 0) and (Vy = 0) and (GunExplosion = 0) then
begin
try PlaySound(PChar('GunShot.wav'), 0, SND_ASYNC);
except
end;
Vx := V * Sin(Angle * pi / 180); Vy := V * Cos(Angle * pi / 180);
x := GunPosition + 15 * Sin(Angle * pi / 180);
y := 20 + 15 * Cos(Angle * pi / 180);

```

```

end;
if (Key = 'Q') then
begin
try PlaySound(PChar('Explode.wav'), 0, SND_ASYNC);
except
end;
for i := 0 to BallCount - 1 do
if Balloons[i].Explosion = 0 then Balloons[i].Explosion := 1;
Caption := 'Уничтожен';
end; end;

```

8. В обработчике изменения положения бегунка *TrackBar* меняем наклон пушки.

```

procedure TForm1.TrackBar1Change(Sender: TObject);
begin
Angle := TrackBar1.Position;
end;

```

Выстрел происходит, если: нажат пробел; снаряд готов ($V_x = 0$) и ($V_y = 0$); пушка боееспособна ($GunExplosion = 0$).

9. Процедура *MoveAll*, вызываемая каждый такт таймера, отвечает за то, чтобы все процессы шли своим чередом. Вставьте ее сразу после implementation.

```

procedure MoveAll;
var i: integer;
begin
for i := 0 to BallCount - 1 do
with Balloons[i] do
if Explosion = 0 then x := x + v else begin
if Explosion < 10 then inc(Explosion);
end;
if (Vx <> 0) or (Vy <> 0) then begin
x := x + Vx * dt; y := y + Vy * dt; Vy := Vy + g * dt;
end;
if BombY < 1000 then
begin
BombY := BombY + BombV * dt; BombV := BombV + g * dt
end;
if (GunExplosion > 0) and (GunExplosion < 10) then inc(GunExplosion);
end;
end;

```

10. Процедура пододвигает все шары, которые живы (у них $Explosion = 0$), в направлении вектора их скорости. Остальные шары, если они находятся в фазе взрыва ($0 < Explosion < 10$), переходят в следующую фазу.

Если снаряд выпущен ($V_x <> 0$) или ($V_y <> 0$), то его координата и скорость изменяются согласно законам природы. Если бомба сброшена ($BombY < 1000$), то она также подчиняется закону всемирного тяготения.

Если танк находится в фазе взрыва ($0 < GunExplosion < 10$), он переходит в следующую фазу.

Ошибки, возникающие в обработчике таймера или процедурах, которые вызываются из обработчика, не останавливают таймер и, значит, через секунду возникают опять. Чтобы их остановить, приходится использовать *Ctrl-Alt-Del...*

В этой процедуре мы работаем с массивом. Одной из самых частых ошибок, которые не так-то просто найти, является выход за границы массива. Для предотвращения подобных ситуаций включите проверку на выход из диапазона допустимых значений (*Project - Options - Compiler - Runtime errors - Range checking*), которая по умолчанию отключена.

Перемещения сами по себе могли бы продолжаться бесконечно. Необходимо проконтролировать:

- не попал ли снаряд в шар, следовательно, взорвать его;
- не уничтожены ли все шары, следовательно, выдать надпись о победе;
- не поравнялся ли шар с пушкой, следовательно, сбросить бомбу;
- не упал ли снаряд и не вышел ли он из зоны контроля, следовательно, перезарядить пушку;
- не упала ли бомба на пушку, следовательно, закончить игру.

Для контроля за передвижениями предусмотрена процедура:

procedure CheckCollisions;

var i, j: integer;

HappyEnd: boolean;

begin

for i := 0 **to** BallCount - 1 **do**

with Balloons[i] **do**

begin

if (Explosion = 0) **then begin**

if (sqr(x - Unit1.x) + sqr(y - Unit1.y) < sqr(BallRadius)) **then**

begin

try PlaySound(PChar('Explode.wav'), 0, SND_ASYNC);

except end;

Explosion := 1; Unit1.x := 0; Unit1.y := 0; Unit1.Vx := 0;

Unit1.Vy := 0;

HappyEnd := (GunExplosion = 0);

for j := 0 **to** BallCount - 1 **do**

HappyEnd := HappyEnd **and** (Balloons[j].Explosion > 0);

if HappyEnd **then** Form1.Caption := 'Победа!';

end;

if (x = GunPosition) **and** (BombY = 1000) **and** (GunExplosion = 0) **then**

begin

BombY := y - BallRadius - 5; BombV := 0;

end; end; end;

if (y < 0) **or** (x < 0) **or** (x > 440) **then**

begin

x := 0; y := 0; Vx := 0; Vy := 0;

end;

if BombY < 10 **then begin**


```

BombY := 1000; GunExplosion := 1;
Try PlaySound(PChar('Explode.wav'), 0, SND_ASYNC);
except end;
Form1.Caption := 'Увы...';
end; end;

```

В этом блоке возможная ошибка связана с использованием **with ... do begin ... end**. X-координата снаряда хранится в глобальной переменной *x*, X-координата воздушного шара – в переменной *Balloons[i].x*. Однако когда пишут код внутри **with Balloons[i] do begin ... end**, то именно к координате шара обращаются просто как к *x*. Если нужно добраться до глобальной переменной, то теперь уже перед ней нужно ставить уточнение: *Unit1.x*, где *Unit1* – имя модуля.

11. Рисование отдельных элементов батальи также разумно поместить в специализированные для этого процедуры.

Художественные образы шаров, пушки и взрыва передаются с помощью процедур:

```

procedure DrawBalloon(x, y: integer; Color: TColor);
begin
with Form1.Image1.Canvas do
begin
Pen.Color := clBlack; Brush.Color := Color;
Ellipse(x - BallRadius, y - BallRadius, x + BallRadius, y + BallRadius);
Pen.Color := clWhite; Brush.Color := clWhite;
Ellipse(x - BallRadius div 2 - 3, y - BallRadius div 2 - 3, x - BallRadius div 2
+ 3, y - BallRadius div 2 + 3);
Pen.Color := clBlack; Brush.Color := clOlive;
Rectangle(x - 5, y + BallRadius + 5, x + 5, y + BallRadius + 10);
MoveTo(x - 5, y + BallRadius + 5);
LineTo(x - BallRadius, y);
MoveTo(x, y + BallRadius + 5);
LineTo(x, y);
MoveTo(x + 5, y + BallRadius + 5);
LineTo(x + BallRadius, y);
end; end;
procedure DrawGun;
begin
with Form1.Image1.Canvas do
begin
if (Vx = 0) and (Vy = 0) then
Pen.Color := RGB(0, 70, 0) else Pen.Color := clBlack;
Brush.Color := Pen.Color; Pen.Width := 5;
MoveTo(GunPosition, 240);
LineTo(GunPosition + round(15 * sin(Angle * pi / 180)), 240 - round(15 *
cos(Angle * pi / 180)));

```

```

Pen.Width := 1;
Ellipse(GunPosition - 8, 232, GunPosition + 8, 248);
Rectangle(GunPosition - 10, 240, GunPosition + 10, 260);
end; end;

```

12. Процедура DrawExplosion обеспечивает доступ к координатам снаряда.

```

procedure DrawExplosion(x, y, Phase: integer);
var i, xx, yy, Size: integer; a, b: double;
begin
with Form1.Image1.Canvas do
for i := 0 to Phase * 10 do
begin
a := random * 2 * pi;
b := random * sqr(Phase) / 3 + 5;
xx := x + round(1 * sin(a));
yy := y + round(1 * cos(a));
Size := round(sqr(10 - Phase) / 8 + 2);
Pen.Color := RGB(random(100), 0, 0);
Brush.Color := Pen.Color;
Ellipse(xx - random(Size) - 1, yy - random(Size) - 1, xx + random(Size), yy +
random(Size));
end; end;

```

13. Кроме того, пушка меняет цвет, когда готов очередной снаряд. В довершение, нужно соединить все написанное воедино.

Процедуры рисования объединяются в процедуру, рисующей все поле боя: небо, шары, пушка, ядро, бомба.

14. Объединяет все вместе обработчик таймера.

```

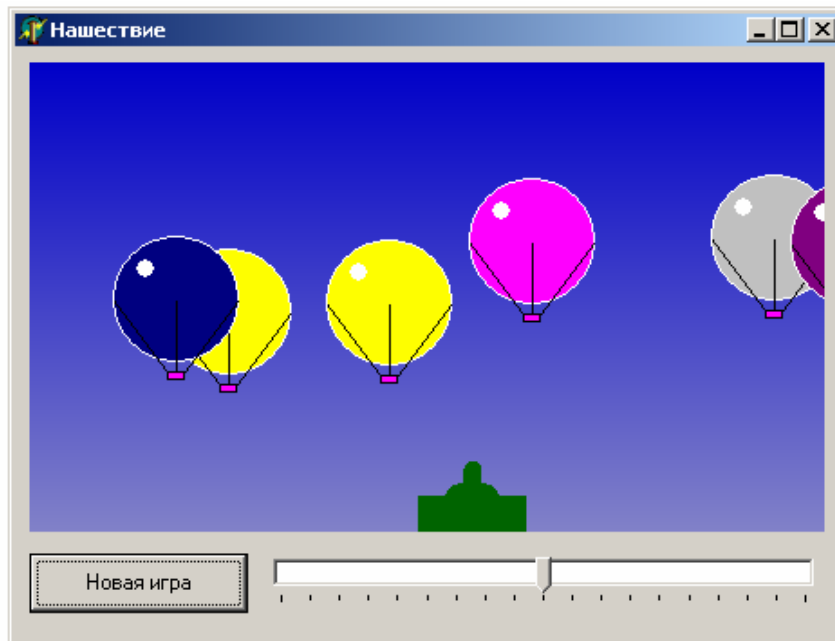
procedure DrawBattleField;
var i: integer;
begin
with Form1.Image1.Canvas do
for i := 0 to 259 do
begin
Pen.Color := RGB(i div 2, i div 2, 255); MoveTo(0, i); LineTo(440, i);
end;
for i := 0 to BallCount - 1 do
with Balloons[i] do
if Explosion = 0 then DrawBalloon(x, 260 - y, Color) else if Explosion < 10
then DrawExplosion(x, 260 - y, Explosion);
if GunExplosion = 0 then DrawGun else if GunExplosion < 10 then
DrawExplosion(GunPosition,
240, GunExplosion);
with Form1.Image1.Canvas do
begin
Pen.Color := clMaroon;
Brush.Color := clRed;

```

```

if (Vx <> 0) or (Vy <> 0) then Ellipse(round(x) - 2, 260 - round(y) - 2,
round(x) + 3, 260 - round(y) + 3);
if (BombY <> 1000) then Ellipse(GunPosition - 2, 260 - round(BombY) - 2,
GunPosition + 3, 260 - round(BombY) + 3);
end; end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
MoveAll; CheckCollisions; DrawBattleField;
end;

```



15. Сохраните проект и проверьте его работоспособность.

ПРАКТИЧЕСКАЯ РАБОТА № 32

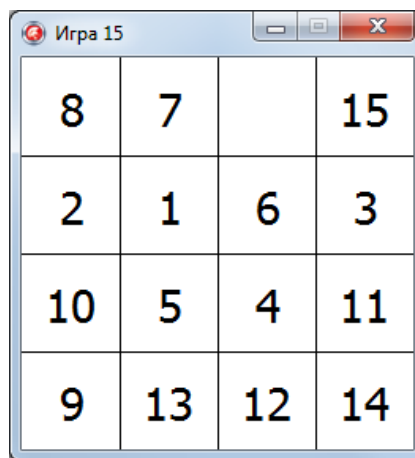
Тема: Разработка игрового приложения

Цель работы: научиться разрабатывать игровые приложения.

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

Содержание работы:

Задание 1. Всем известна игра "15". Вот ее правила. В прямоугольной коробочке находятся 15 фишек, на которых написаны числа от 1 до 15. Размер коробочки — 4×4, таким образом в коробочке есть одна пустая ячейка. В начале игры фишки перемешаны. Задача игрока состоит в том, чтобы, не вынимая фишки из коробочки, выстроить фишки в правильном порядке. Напишите программу **Игра 15**.



8	7		15
2	1	6	3
10	5	4	11
9	13	12	14

```
unit game15;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,  
Dialogs;
```

```
type
```

```
TForm1 = class(TForm)
```

```
  procedure FormCreate(Sender: TObject);
```

```
  procedure FormMouseDown(Sender: TObject; Button: TMouseButton; Shift:  
    TShiftState; X, Y: Integer);
```

```
  procedure FormPaint(Sender: TObject);
```

```
private { Private declarations }
```

```
// эти объявления вставлены сюда вручную
```

```
  procedure ShowPole;
```

```
  procedure Mixer;
```

```
public { Public declarations }
```

```
end;
```

```
var
```

```
Form1: TForm1; implementation
```

```
{ $R *.dfm }
```

```
const H = 4; W = 4; // размер поля — 4x4
```

```

CH = 62; CW = 62; // размер клеток — 16x16
var
// правильное расположение фишек
stp : array[1..H, 1..W] of byte = (( 1, 2, 3, 4), ( 5, 6, 7, 8), ( 9,10,11,12), (13,14,15,
0));
// игровое поле
pole: array[1..H, 1..W] of byte;
ex,ey: integer; // координаты пустой клетки
procedure NewGame; // новая игра
var i,j: integer;
begin
// исходное (правильное) положение
for i:=1 to H do
for j:=1 to W do
pole[i,j] := stp[i,j]; Form1.Mixer; // перемешать фишки
Form1.ShowPole; // отобразить поле
end;
// проверяет, расположены ли фишки в нужном порядке
function Finish: boolean;
var row,col: integer; i: integer;
begin
row :=1; col :=1;
Finish:= True; // пусть фишки в нужном порядке
for i:=1 to 15 do
begin
if pole[row,col] <> i then begin
Finish:= False; break;
end;
// к следующей клетке
if col < 4 then inc(col) else begin
col :=1; inc(row);
end; end; end;
// "перемещает" фишку в соседнюю пустую клетку, если она есть, конечно
procedure Move(cx,cy: integer);
// cx,cy — клетка, в которой игрок сделал щелчок
var r: integer; // выбор игрока
begin
// проверим, возможен ли обмен
if not (( abs(cx-ex) = 1) and (cy-ey = 0) or ( abs(cy-ey) = 1) and (cx-ex = 0))
then exit;
// обмен: переместим фишку из x,y в ex,ey
Pole[ey,ex] := Pole[cy,cx]; Pole[cy,cx] := 0; ex:=cx; ey:=cy;
// отрисовать поле
Form1.ShowPole;
if Finish then

```

```

begin
r := MessageDlg('Цель достигнута!' + #13 + 'Еще раз?', mtInformation,
[mbYes, mbNo], 0);
if r = mrNo then Form1.Close; // завершить работу программы
end; end;
// щелчок в клетке
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
var cx, cy: integer; // координаты клетки
begin
// преобразуем координаты мыши в координаты клетки
cx := Trunc(X / CW) + 1; cy := Trunc(Y / CH) + 1; Move(cx, cy);
end;
procedure TForm1.ShowPole; // выводит игровое поле
var i, j: integer;
x, y: integer; // x, y — координаты вывода текста в клетке
begin
// сетка: вертикальные линии
for i:= 1 to W — 1 do
begin
Canvas.MoveTo(i*CW, 0); Canvas.LineTo(i*CW, ClientHeight);
end;
// сетка: горизонтальные линии
for i:= 1 to H — 1 do
begin
Canvas.MoveTo(0, i*CH); Canvas.LineTo(ClientWidth, i*CH);
end; // содержимое клеток x, y — координаты вывода текста
for i:= 1 to H do
begin y:=(i-1)*CH + 15;
for j:=1 to W do
begin x:=(j-1)*CW + 15;
case Pole[i, j] of
0: Canvas.TextOut(x, y, ' ');
1..9: Canvas.TextOut(x, y, ' ' + IntToStr(Pole[i, j]) + ' ');
10..15: Canvas.TextOut(x, y, IntToStr(Pole[i, j]));
end; end; end; end;
// "перемешивает" фишки
procedure TForm1.Mixer;
var x1, y1: integer; // пустая клетка
x2, y2: integer; // эту переместить в пустую
d: integer; // направление, относительно пустой
i: integer;
begin
x1:=4; y1:=4;
randomize;

```

```

for i:= 1 to 150 do
begin
repeat
x2:=x1; y2:=y1; d:=random(4)+1;
case d of
1: dec(x2);
2: inc(x2);
3: dec(y2);
4: inc(y2);
end;
until (x2>=1) and (x2<=4) and (y2>=1) and (y2<=4);
// здесь определили фишку, которую надо переместить в пустую клетку
Pole[y1,x1] := Pole[y2,x2]; Pole[y2,x2] := 0; x1:=x2; y1:=y2; end;
// запомним координаты пустой клетки
ex:= x1; ey:= y1; end;
// обработка события Create
procedure TForm1.FormCreate(Sender: TObject);
begin
ClientWidth := CW * W; ClientHeight := CH * H;
Canvas.Font.Name := 'Times New Roman'; Canvas.Font.Size := 22; NewGame;
end;
// обработка события Paint
procedure TForm1.FormPaint(Sender: TObject);
begin
Form1.ShowPole;
end; end.

```

ПРАКТИЧЕСКАЯ РАБОТА № 33

Тема: Разработка многооконного приложения. Компиляция и запуск приложения

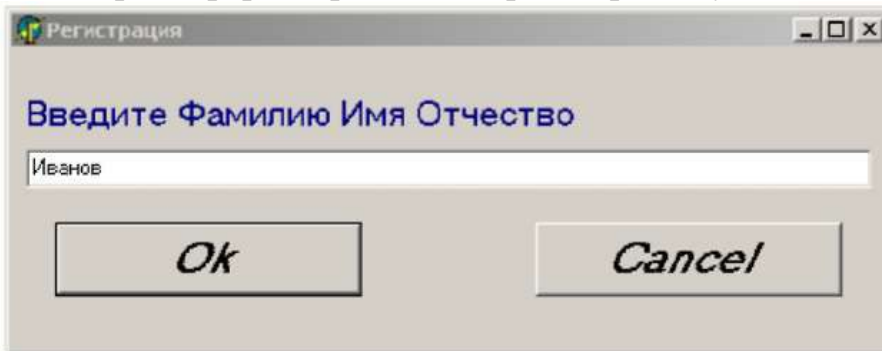
Цель работы: научить создавать многооконное приложение, отработать компиляцию и запуск приложения

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

Содержание работы:

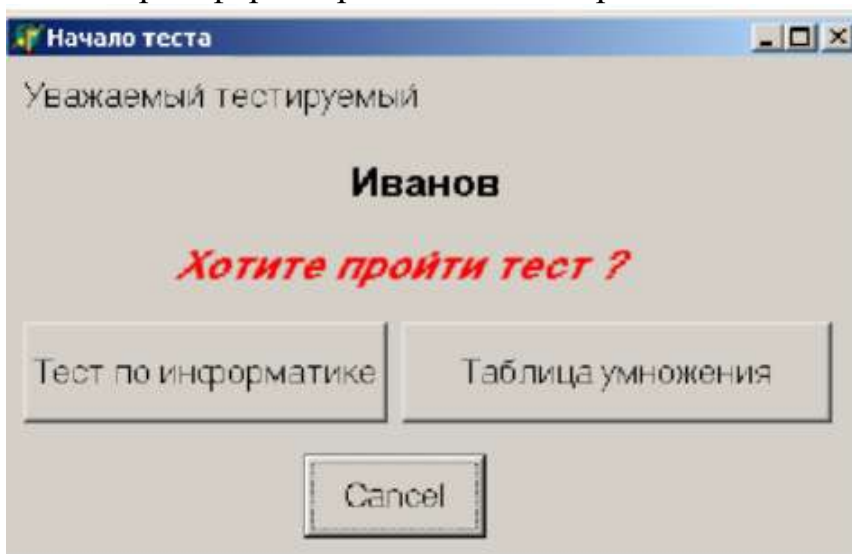
Задание 1. Создать программу, которая тестирует учащегося по информатике и математике. Проект должен содержать последовательность форм, реализующих диалог с тестируемым учащимся.

1. Запустите интегрированную среду разработчика Borland Delphi 7. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её *Тест*. Сохраните проект в созданную ранее папку, выполнив команду *File - Save Project as...*
3. На первой форме происходит регистрация учащегося:



Фрагмент программы (unit1):
uses Unit2; {\$R *.dfm}
procedure TForm1.Button2Click(Sender: TObject); begin Close; end;
procedure TForm1.Button1Click(Sender: TObject); begin
Form2.Label3.Caption:=Form1.Edit1.Text; Form2.ShowModal; end;

4. На второй форме предлагается выбрать один из тестов.



Фрагмент программы (unit2):
uses Unit3, Unit6; {\$R *.dfm}


```

procedure TForm2.Button1Click(Sender: TObject);
begin Form3.ShowModal; end;
procedure TForm2.Button2Click(Sender: TObject);
begin Form2.Close; end;
procedure TForm2.Button3Click(Sender: TObject);
begin Form6.Edit2.Text:="";
Form6.ShowModal;
end;

```

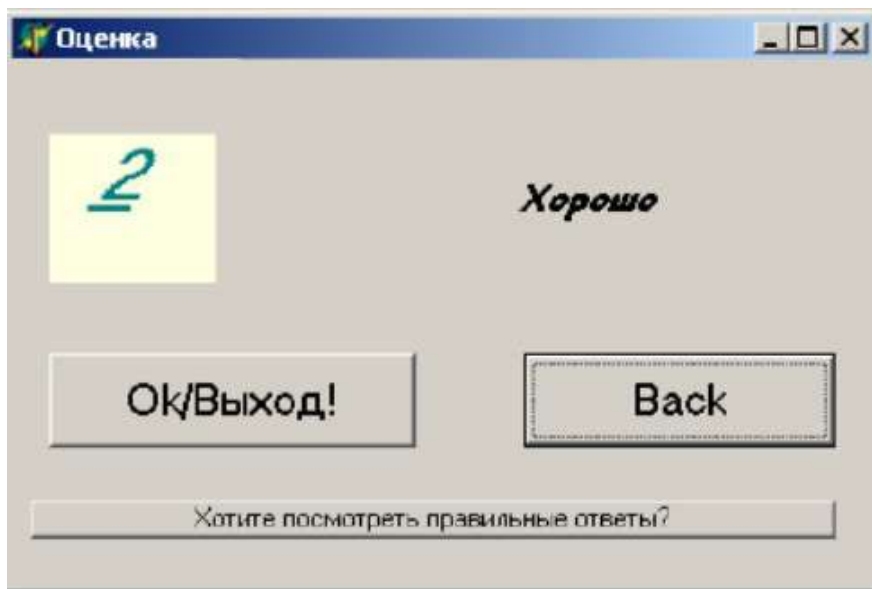
5. На третьей форме предлагается пройти тест по информатике:

```

Фрагмент программы (unit3): uses Unit4, Unit2; {$R *.dfm}
procedure TForm3.Button1Click(Sender: TObject);
begin
k:=0;
if (Form3.Edit1.Text='монитор') or (Form3.Edit1.Text='Монитор') or
(Form3.Edit1.Text='МОНИТОР') then k:=k+1;
if (Form3.Edit2.Text='Клавиатура') or (Form3.Edit2.Text='клавиатура')
or (Form3.Edit2.Text='КЛАВИАТУРА') then k:=k+1; if Form3.Edit3.Text='8'
then k:=k+1; Form4.Label2.Caption:=IntToStr(k); if k=0 then
Form4.Label1.Caption:='Очень плохо' else
if k=1 then Form4.Label1.Caption:='Плохо' else
if k=2 then Form4.Label1.Caption:='Хорошо' else
if k=3 then Form4.Label1.Caption:='Очень хорошо';
Form3.Edit1.Text:="";
Form3.Edit2.Text:="";
Form3.Edit3.Text:="";
Form4.ShowModal;
end;
procedure TForm3.Button2Click(Sender: TObject); begin
Form3.Close; end;

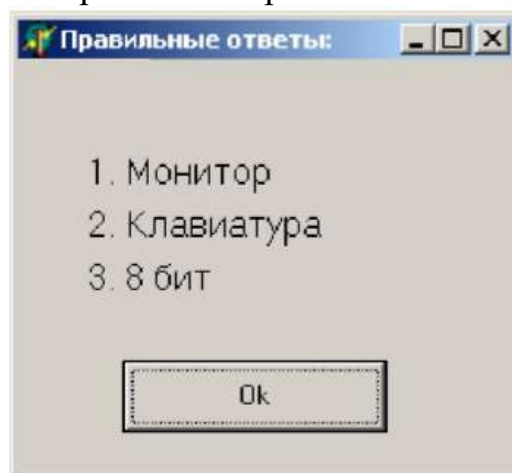
```

6. На следующей форме отображается результат тестирования и предложение о просмотре ответа.



Фрагмент программы (unit4): uses Unit1, Unit5, Unit3, Unit2; {\$R *.dfm}
 procedure TForm4.Button2Click(Sender: TObject);
 begin Form4.Close; end;
 procedure TForm4.Button1Click(Sender: TObject);
 begin Form4.Close; Form3.Close; end;
 procedure TForm4.Button3Click(Sender: TObject);
 begin Form5.ShowModal; end;

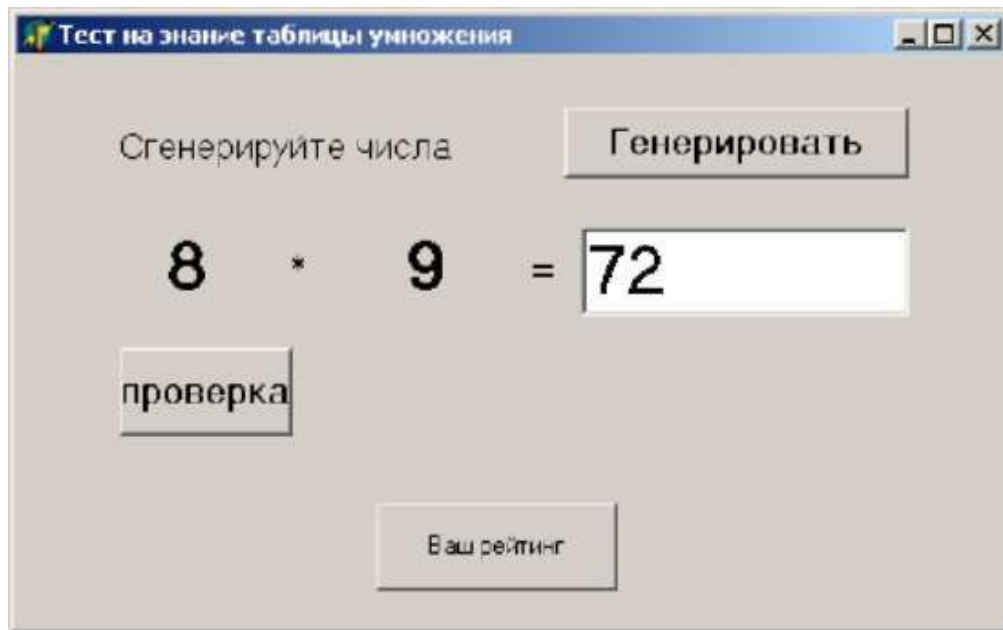
7. На следующей форме отображаются правильные ответы.



Фрагмент программы (unit5):
 uses Unit4, Unit3, Unit2, Unit1;
 {\$R *.dfm}
 procedure TForm5.Button1Click(Sender: TObject); begin
 Form4.Close; Form3.Close; Form2.Close; Form1.Close;
 Form5.Close; end;

8. Если учащийся выбрал тест по математике, то ему предлагается проверить свои знания таблицы умножения. На следующей форме случайным образом выбираются числа. Учащийся должен ввести значение произведения в текстовое поле. С помощью кнопки "проверка" выясняется правильность введенного ответа. Если ответ правильный, то можно сгенерировать

следующий пример. После нескольких примеров можно проверить свой рейтинг.

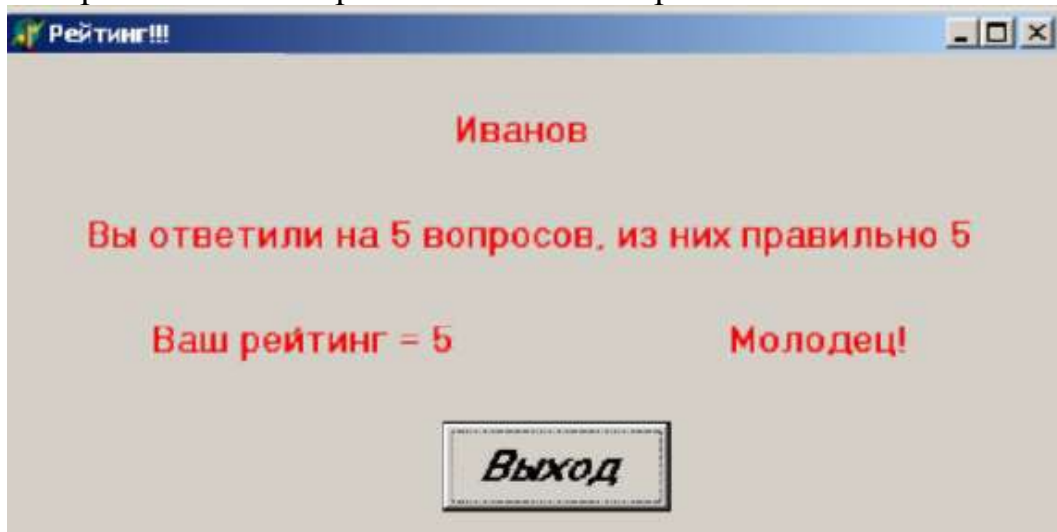


Фрагмент программы (unit6):

```
uses Unit7, Unit1; {$R *.dfm}
procedure TForm6.Button1Click(Sender: TObject);
var n,i:integer;
begin
randomize;
a:=random(10)-0; b:=random(10)-0;
Form6.Label1.Caption:=IntToStr(a);      Form6.Label2.Caption:=IntToStr(b);
Form6.Edit2.Text:='-'; Form6.Label6.Caption:=''; end;
procedure TForm6.Button2Click(Sender: TObject);
begin
if (a*b=StrToInt(Form6.Edit2.Text)) then begin
Form6.Label6.Caption:='Правильно';
m:=m+1; r:=r+1; q:=q+1; e
nd else begin
Form6.Label6.Caption:='Не правильно'; r:=r-1;
q:=q+1; end;
Form6.Label1.Caption:=''; Form6.Label2.Caption:='';
end;
procedure TForm6.Button3Click(Sender: TObject);
var c:real;
begin
Form7.Label2.Caption:=Form1.Edit1.Text;
Form7.Label1.Caption:='Вы ответили на '+IntToStr(q)+' вопросов, из них
правильно '+IntToStr(m);
Form7.Label4.Caption:='Ваш рейтинг = '+IntToStr(r); c:=m/q;
if c=0 then Form7.Label3.Caption:='Очень плохо' else if (c>0)and(c<0.5) then
Form7.Label3.Caption:='Плохо'           else           if           c=0.5           then
```

```
Form7.Label3.Caption:='Надо доучить' else if (c>0.5) and (c<1) then  
Form7.Label3.Caption:='Хорошо' else if c=1 then  
Form7.Label3.Caption:='Молодец!'; Form7.ShowModal; end;
```

9. При нажатии на кнопку "Ваш рейтинг" на следующей форме появляются результаты рейтинга. Тестирование можно завершить.



Фрагмент программы (unit7):

```
uses Unit6; {$R *.dfm}  
procedure TForm7.Button1Click(Sender: TObject);  
begin  
Form7.Close; Form6.Close; end;
```

Обратите внимание на подключение модулей в строке Uses. Таким образом, происходит обращение к соответствующей форме. Для отображения формы используется функция *function ShowModal: Integer;*

Данная функция позволяет показывать форму в работе режима диалога.

10. Выполните программу, щелкнув *Run* (выполнить) на панели *Отладка* или нажав *F9*. В случае появления ошибок, исправьте их. Сохраните проект.

ПРАКТИЧЕСКАЯ РАБОТА № 34

Тема: Разработка многооконного приложения. Компиляция и запуск приложения

Цель работы: научить создавать многооконное приложение, отработать компиляцию и запуск приложения

Оборудование: ПК, программное обеспечение – Borland Delphi, инструкции по выполнению работы

Содержание работы:

Задание 1.Разработать многооконное приложение по индивидуальным заданиям. Произвести его компиляцию и запуск. Проверить работоспособность приложения.

Информационное обеспечение обучения по дисциплине

Основные учебные издания:

1. Дорохова, Т. Ю. Основы алгоритмизации и программирования: учебное пособие для СПО / Т. Ю. Дорохова, И. Е. Ильина. — Саратов, Москва: Профобразование, Ай Пи Ар Медиа, 2022. — 139 с. — ISBN 978-5-4488-1531-7, 978-5-4497-1718-4. — Текст: электронный // Цифровой образовательный ресурс IPR SMART: [сайт]. — URL: <https://profspo.ru/books/122426>
2. Золин, А. Г. Программирование на C++: учебное пособие для СПО / А. Г. Золин, А. Е. Колоденкова, Е. А. Халикова. — Саратов: Профобразование, 2022. — 126 с. — ISBN 978-5-4488-1439-6. — Текст: электронный // ЭБС PROФобразование: [сайт]. — URL: <https://profspo.ru/books/116283>
3. Лебеденко, Л. Ф. Технологии программирования: учебно-методическое для СПО / Л. Ф. Лебеденко, О. И. Моренкова. — Саратов: Профобразование, 2021. — 108 с. — ISBN 978-5-4488-1204-0. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/106637>
4. Моренкова, О. И. Программирование на языке C/C++: практикум для СПО / О. И. Моренкова, Т. И. Парначева. — Саратов: Профобразование, 2021. — 102 с. — ISBN 978-5-4488-1192-0. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/106631>

Дополнительные учебные издания:

5. Кудинов, Ю. И. Основы алгоритмизации и программирования: учебное пособие для СПО / Ю. И. Кудинов, А. Ю. Келина. — 2-е изд. — Липецк, Саратов: Липецкий государственный технический университет, Профобразование, 2020. — 71 с. — ISBN 978-5-88247-956-4, 978-5-4488-0757-2. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/92834>
6. Токманцев, Т. Б. Алгоритмические языки и программирование: учебное пособие для СПО / Т. Б. Токманцев; под редакцией В. Б. Костоусова. — 2-е изд. — Саратов, Екатеринбург: Профобразование, Уральский федеральный университет, 2019. — 102 с. — ISBN 978-5-4488-0510-3, 978-5-7996-2899-4. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/87785>

Электронно-библиотечная система:

7. ЭБС «IPRbooks», ООО «Ай Пи Ар Медиа»
8. ЭБС «PROФобразование»
9. ЭБС «Book.ru»