Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Тульский государственный университет

Институт прикладной математики и компьютерных наук

Кафедра информационной безопасности

Направление: 09.03.03 Прикладная информатика

Дисциплина: Программирование

Методические указания к лабораторным работам

Тула, 2017 год

Лабораторная работа №1
Знакомство со средой программирования
Microsoft Visual C++

1. ЦЕЛЬ РАБОТЫ: Изучение меню и настроек интегрированной среды Visual C++.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Среда программирования Microsoft Visual C++ входит в состав пакета Microsoft Visual Studio, а также поставляется в виде отдельного инструмента.

Visual Studio Express Editions – это серия инструментов семейства Visual Studio, представляющие собой упрощенную и облегченную среду разработки с усеченными возможностями профессиональных версий Visual Studio.

Например, Visual Studio 2008 Express Editions содержит следующие инструменты:

- Visual C# 2008 Express Edition;
- Visual Basic 2008 Express Edition;
- Visual C++ 2008 Express Edition;
- Visual Web Developer 2008 Express Edition — облегченный инструмент для создания динамически-обновляемых веб-сайтов и веб-сервисов.

Среда программирования Visual C++ состоит из следующих основных компонентов:

- редактор исходного текста;
- редактор ресурсов;
- компилятор кода;
- компилятор ресурсов – компилирует текстовые файлы с описанием ресурсов (RS) в двоичные RES – файлы;
- компоновщик – служит для формирования исполняемого Exe файла;
- отладчик – выполняет трассировку программы (пошаговое выполнение) с целью поиска ошибок в программе.

В связи с тем, что уровень сложности программного обеспечения очень высок, разработка приложений Windows с использованием только какого-либо языка программирования (например, языка C) значительно затрудняется. Программист должен затратить массу времени на решение стандартных задач по созданию многооконного интерфейса. Чтобы облегчить работу программиста практически все современные компиляторы языка C++ содержат специальные библиотеки классов. Такие библиотеки включают в себя практически весь программный интерфейс Windows и позволяют пользоваться при программировании средствами более высокого уровня, чем обычные вызовы функций. За счет этого значительно упрощается разработка приложений, имеющих сложный интерфейс пользователя, облегчается поддержка технологии OLE и взаимодействие с базами данных.

Библиотеки Visual C++:

1) Standard C++ Library;
2) библиотека классов Microsoft Foundation Class Library (MFC);
3) библиотека классов.Net Framework –;

4) Microsoft Active Template Library (ATL) – представляет собой средство построения элементов управления ActiveX.

5) Библиотека C Run-Time Library (CRT)

6) OLE DB Templates и др.

Одной из самых популярных библиотек C++ является библиотека классов MFC (Microsoft Foundation Class).

MFC – это базовый набор (библиотека) классов, написанных на языке C++ и предназначенных для упрощения и ускорения процесса программирования под Windows.

Библиотека содержит многоуровневую иерархию классов, которые дают возможность создавать Windows-приложения на базе объектно-ориентированного подхода.

В Visual C++ могут быть встроены средства, позволяющие программисту облегчить разработку приложений. В первую очередь к ним относятся MFC AppWisard, ClassWizard и редактор ресурсов.

Мастер приложений (AppWizard) – генератор кода, создающий рабочую заготовку Windows-приложения с теми компонентами, именами классов, которые программист задает в его диалоговых окнах. Конечно, MFC AppWizard не всесилен. Прикладную часть приложения программисту придется разрабатывать самостоятельно. Исходный текст приложения, созданный MFC AppWizard, станет только основой, к которой нужно подключить остальное. Но работающий шаблон приложения – это уже половина всей работы. Исходные тексты приложений, автоматически полученных от MFC AppWizard, могут составлять сотни строк текста. Набор его вручную был бы очень утомителен.

Мастер классов (ClassWizard) – программа, реализованная как DLL (сокращ. от англ. Dynamic-Link Library – динамически подключаемая библиотека); избавляет программистов от монотонной работы, связанной с кодированием классов Visual C++ (с созданием новых классов, виртуальных функций и т.п.);

Консольные и оконные приложения

В среде программирования Visual C++ можно создавать консольные и оконные приложения.

Консоль — это монитор и клавиатура. Консольное приложение – приложение, не имеющее графического интерфейса, для которого устройством ввода является клавиатура, а устройством вывода — монитор. Такие программы работают в окнах, напоминающих окна сеансов DOS (рисунок 1).

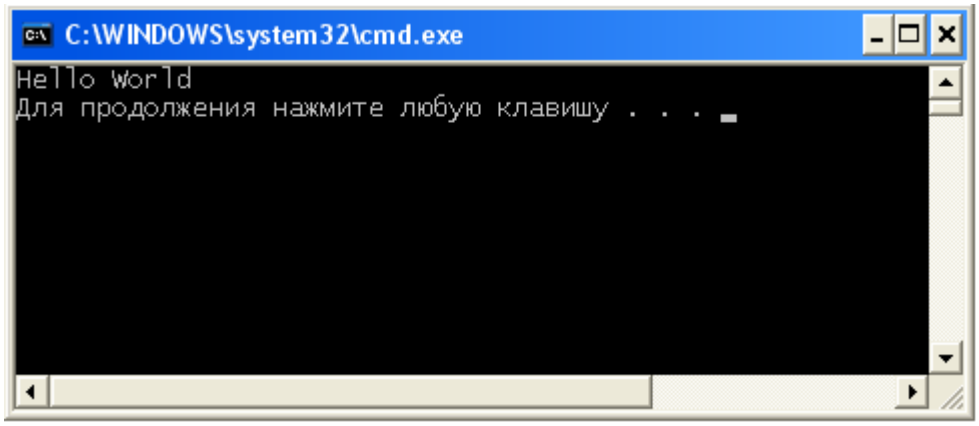


Рисунок 1 – Консольное приложение

Консольные приложения удобны как иллюстрации при рассмотрении общих вопросов программирования, когда необходимо сосредоточиться на сути проблемы не задумываясь о пользовательском интерфейсе. На практике консольные приложения применяются для различного вида утилит, тестовых программ и т.д.

Оконное приложение (Windows-приложение) – приложение, в котором используется Windows-интерфейс GUI (Grphical User Interface – графический интерфейс пользователя).

Отправной точкой любой разработки в Visual C++ является диалоговое окно File→New→Project. Для создания консольного приложения следует выбрать CLR Console Application.

CLR, сокр. от Common Language Runtime – «общеязыковая среда выполнения» — это компонент пакета Microsoft .NET Framework, виртуальная машина, на которой исполняются все языки платформы .NET Framework.

Компиляция программы

Чтобы скомпилировать программу в интегрированной среде разработки (Integrated Development Environment — IDE) Visual C++ необходимо выбрать команду Start Debugging с панели инструментов, меню или клавишей F5 (рисунок 2).

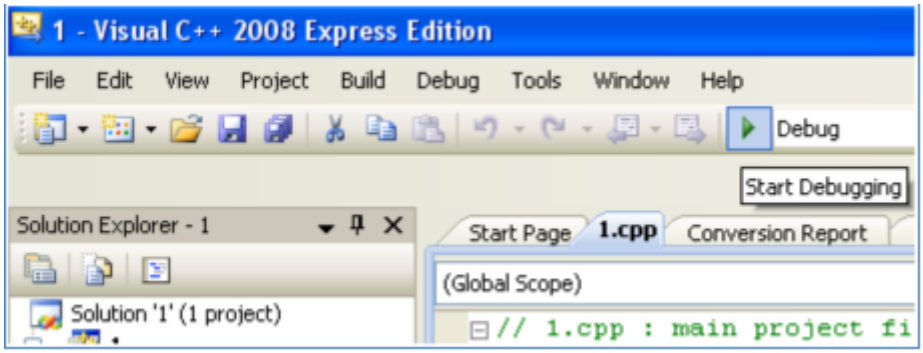
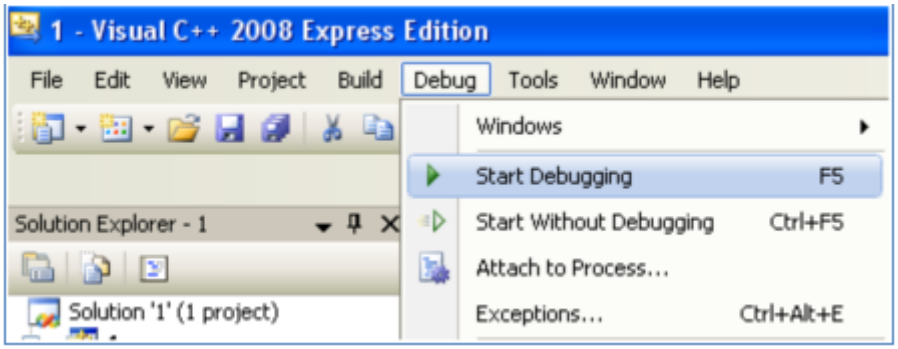


Рисунок 2 – Компиляция программы



Продолжение рисунка 2

В результате работы C++-компилятора получается выполняемый объектный код. Для Windows-среды выполняемый файл будет иметь то же имя, что и исходный, но другое расширение, а именно расширение .exe.

При попытке скомпилировать программу с ошибками компилятор выдаст сообщение о наличии синтаксических ошибок и укажет их количество (рисунок 3).

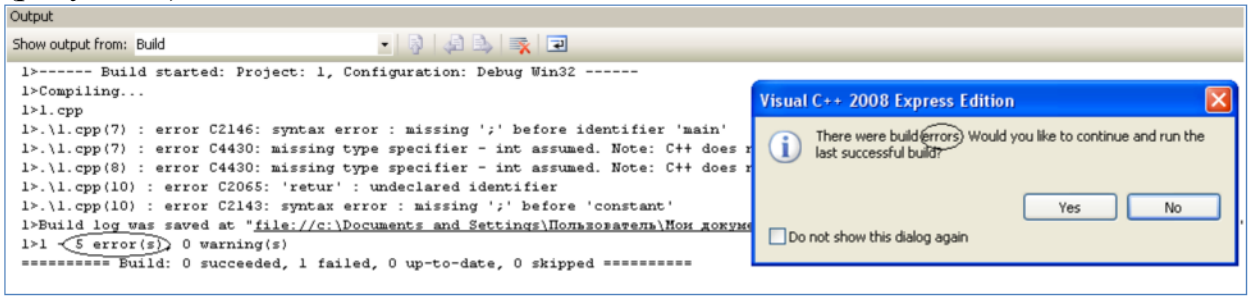


Рисунок 3 – Сообщение о наличии ошибок

Если компилятор «подозревает», но не «уверен» в ошибке (например, некорректность конструкции при видимой правильности с точки зрения синтаксиса), то он выдает предупреждение warning(s).

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей, клавиатура, мышь, среда Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Установить среду программирования Visual C++ и изучить ее интерфейс.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения
2. Запустить Visual C++ и создать консольное приложение.
3. Запустить программу на выполнение (сочетание клавиш Ctrl+F5)
4. Внести в программу ошибки
5. Скомпилировать программу и проанализировать результат
6. Оформить отчет

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Текст программы
4. Результаты и выводы по лабораторной работе

Лабораторная работа №2
Основы языка С/С++. Линейные программы

1. ЦЕЛЬ РАБОТЫ

Изучение меню и настройка интегрированной среды. Получение навыков по составлению и отладке простейших программ на языке Си. Изучение организации простейшего ввода-вывода на языке Си.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Структура программы на языке Си

Общая структура программы на Си содержит следующие разделы:

# include <aa1> … # include <aaN>	Раздел подключения библиотек. aa1, aa2,… aaN - имена библиотек
# define ип*1*[1] значение1 … # define ип*M* значениеM	Раздел описания констант. ип1,…, ип*M* – константы.
Тип1 ип*1*,…,ип*K*;	Раздел описания глобальных переменных
…	
ТипM1 ип*1*,…,ип*K1*;	
Тип_ф-ции ИФ[2] (Тип1 ип*1*,…,ип*K*);	Раздел объявления функций
main()	
{	
Тип1 ип*1*,…,ип*K*;	Раздел описания локальных переменных
…	
ТипM ип*1*,…,ип*K1*;	
/* Тело функции main */	
return();	
}	
Тип_ф-ции ИФ (Тип1 ип*1*,…,ип*K*)	Раздел определения функций
{	
return();	
}	

Переменные и типы языка Си

В языке Си все переменные должны быть описаны до их использования, обычно это делается в начале функции до первого выполняемого оператора.

Описание состоит из спецификатора типа и списка переменных, имеющих этот тип.

Тип int означает, что все переменные списка целые, т.е. числа, лежащие между -32768 и +32767.

Тип float предназначен для чисел с плавающей точкой, т.е. для чисел, которые могут иметь дробную часть и лежащие в диапазоне от 10e-38 до 10e+38.

Тип char - символ - один байт.

Тип double плавающее с двойной точностью (для 16-разрядных ПЭВМ 64 бита со знаком).

С типом можно использовать квалификаторы (модификаторы):

short (короткое),

long (длинное)

---

[1] ип – имя переменной

[2] ИФ – имя функции

unsigned (без знака).

Функция printf()

Функция printf предназначена для вывода информации на экран. Формат ее прост и гибок:

printf("<форматная строка>",<аргумент>,<аргумент>,...);

Форматная строка - это строка, которая начинается и заканчивается двойными кавычками (" "). Целью функции printf является вывод этой строки на экран. До вывода аргументы, указанные в строке, преобразуются в соответствии со спецификаторами формата, указанными в форматной строке.

Для каждого спецификатора формата должен быть указан один и только один аргумент. Если аргумент принадлежит типу данных, который не соответствует указанному в спецификаторе формата, то Турбо-Си попытается проделать соответствующее преобразование. Аргументами могут быть переменные, константы, выражения, вызовы функций, т.к. они могут быть чем угодно, возвращающим значение, подходящее соответствующей спецификации формата.

Наиболее часто используются следующие спецификаторы форматов:

%d - целое

%u - беззнаковое целое

%p - значение указателя

%f - с плавающей точкой

%e - с плавающей точкой в экспоненциальной форме

%c - символ

%s - строка

%x или %X целое в шестнадцатеричном формате

Ширина поля может быть может быть указана цифрой, помещенной между знаком % и буквой. Например, десятичное поле шириной 4 символа будет указано так: %4d. Значение будет напечатано с выравниванием вправо (с ведущими нулями), так что общая длина поля будет равна 4 символам.

Если вам нужно напечатать знак процента, вставьте %%.

Пример. printf("значение переменной a =,%d\n");

Последовательность \n не является спецификатором формата. По традиции это называется esc-последовательностью и представляет собой символ перехода на новую строку, поэтому после печати строки маркер переходит в начало следующей строки.

Если нужно напечатать обратную косую черту - напечатайте две косых черты \\.

Функция scanf()

Основное назначение функции scanf - интерактивный ввод информации.

Формат функции scanf: scanf ("<форматная строка>",<адрес>,<адрес>,...);

Функция scanf использует многие из тех спецификаторов формата со знаком %, что и функция printf.

Однако, у этой функции есть важная отличительная черта: за форматной строкой следуют не значения, а адреса.

Например, следующая строка: scanf("%d%d",*&a*,*&b*) означает, что программа ожидает ввода двух десятичных целых значений, разделенных пробелом; первое значение будет присвоено переменной *a*, второе - переменной *b*.

Если вместо пробела вы хотите разделять свои переменные запятой, отредактируйте эту строку так: scanf("%d,%d",*&a*,*&b*). Такая строка позволит передавать в программу значения, разделяя их запятой.

Функция gets()

С помощью функции gets выполняется считывание символов из стандартного входного потока. Если входной поток прерывается символом перехода на новую строку ‘\n’, то этот символ отбрасывается и не попадает в строку s.

Формат функции gets:

gets(<аргумент – строка s>);

Составление и отладка программ

В общем случае для создания и отладки программ необходимо выполнить следующие этапы:

- ввод и редактирование исходной программы;
- запись программы на диск;
- трансляция программы;
- компоновка программы;
- выполнение программы в среде Visual Studio C++.

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда GNC, Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Ознакомиться с теоретическими положениями лабораторной работы. Набрать в редакторе Visual Studio C++ приведенную ниже программу. Сохранить набранную программу на соответствующем диске, в личном каталоге (имя каталога – номер группы). Выполнить необходимые настройки меню системы.

Отладить, откомпилировать программу (создать объектный файл) и выполнить программу.

```
/*          Моя первая программа на Си          */
# include <stdio.h>
main()
{ int x;
   float y, z;
printf("Введите значение переменной x: ");
scanf (“%d”, &x);
printf("Введите значение переменной y: ");
scanf (“%f”, &y);
z=x+y;
```

```
printf("Сумма введенных переменных=%f", z);
z=x*y;
printf("Произведение введенных переменных=%7.3f", z);
return(0);
}
```

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Ввести представленную выше программу с клавиатуры.
3. Отладить программу. Результаты работы программы показать преподавателю.
4. Оформить отчет.
5. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Текст программы
4. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие символы входят в алфавит языка Си?
2. Какова структура программы на языке Си?
3. Как создать новый файл?
4. Как сохранить программу на внешнем носителе?
5. Как запустить программу на выполнение?
6. Как просмотреть результат выполнения программы?
7. Что выделяется значками /* */?
8. Что выполняется в следующей строке - # include <stdio.h>?
9. Для чего используется оператор - printf()?
10. Для чего необходима запись - main()?
11. Для чего необходимы { }?
12. Для чего используется оператор return(0)?
13. Что означает выражение stdio.h?
14. Какие операторы ввода Вы знаете?
15. Какие операторы вывода Вы знаете?
16. В чем особенность оператора scanf?
17. В чем особенность оператора printf?
18. В чем особенность оператора gets?
19. Какие спецификаторы необходимо использовать при работе с целыми числами?
20. Какие спецификаторы необходимо использовать при работе с вещественными числами?
21. Какие спецификаторы необходимо использовать при работе с отдельными символами?
22. Какие спецификаторы необходимо использовать при работе со строками символов?

Лабораторная работа №3
Язык C/C++. Условный оператор

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является получение навыков использования управляющих условных операторов if, if-else, if-else-if, switch и получение навыков использования условного выражения.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Оператор if

Простейший вид оператора if следующий:

```
if (выражение) действие;
```

Если значение выражение истинно, то выполняется действие и программа продолжается, начиная с оператора, следующего за этим действием.

Оператор if, связанный с блоком выполняемых операторов, выглядит следующим образом:

```
if (выражение)
{
действие 1;
действие 2;
…
действие N;
}
```

Оператор if-else

Оператор if-else нужен для того, чтобы программа выполнила два разных действия в зависимости от истинности некоторого выражения. В простейшем случае оператор if-else выглядит следующим образом:

```
if (выражение)
действие 1;
else
действие 2;
```

В этом операторе, если выражение истинно, выполняется действие 1; если же выражение ложно, выполняется действие 2.

Любое из действий или оба вместе могут быть составным блоком, заключенным в фигурные скобки.

Оператор if-else-if

Комбинация операторов if-else-if часто используется для выполнения многочисленных последовательных сравнений. В общем виде они выглядят следующим образом:

```
if (выражение1)
действие 1;
else if (выражение2)
действие 2;
…
else if (выражение N)
действие n;
```

Каждое действие может быть составным блоком в фигурных скобках (причем после закрывающей фигурной скобки точка с запятой не ставится).

Условное выражение ? :

Условное выражение ?: позволяет кратко записать условие проверки. Этот оператор имеет следующий формат:

выражение-условие ? действие 1 : действие 2;

В этом операторе, если выражение-условие истинно, то выполняется действие 1; если же выражение-условие ложно - выполняется действие 2.

Оператор Switch

Часто необходимо сравнить некоторую переменную или выражение с несколькими значениями. Для этого можно использовать оператор switch.

```
switch (выражение)
{
case константа1: операторы1; break;
case константа2: операторы2; break;
⋮
case константа N: операторы N; break;
default: операторы;
}
```

Пример. Произвести выбор режима (1 или 2). Если выбран режим 1 - вычислить y=x+1. Если же выбран режим 2 - вычислить y=x-1. Если же случайно выбрали другую цифру, то вывести на экран соответствующее сообщение.

```
# include <stdio.h>
main()
{
int x,y,r;
x=1;
scanf("%d",&r);
switch (r) {
case 1: y=x+1; break;
case 2: y=x-1; break;
default: printf("Введена цифра %d",r);
```

}
}

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда GNC, Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Составить программу (в соответствии с вариантом задания), работающую в двух режимах: в первом режиме производится вычисление функции M целого типа, во втором - функции Y вещественного типа. Выбор режима осуществить оператором Switch.

Исходные данные задаются с учетом типов переменных: A, B, C - целого типа, X, K - вещественного.

Варианты:

M=A+B+C, где A,B,C – год, месяц и число рождения студента соответственно.

Y=(X*K)/(X-K), где X – отношение даты рождения к месяцу рождения студента; K – отношение месяца рождения студента к году рождения студента.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Получить вариант задания у преподавателя.
3. Разработать схему программы.
4. Составить программу.
5. Отладить программу. Результаты работы программы показать преподавателю.
6. Оформить отчет.
7. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы
5. Текст программы
6. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем особенность работы оператора if?
2. В чем особенность работы оператора if-else?
3. В чем особенность работы оператора if-else- if?
4. Какой тип может иметь константа в операторе Switch?
5. В чем особенность работы оператора Switch?
6. В чем особенность использования оператора break в работе оператора Switch?
7. Когда можно использовать условное выражение?
8. В чем особенность использования условного выражения?

Лабораторная работа №4
Язык C/C++. Операторы циклов

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является освоение навыков работы с операторами цикла FOR, WHILE, DO-WHILE.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Цикл while

Цикл while является наиболее общим видом цикла, он может использоваться для замены других циклов. Другими словами, цикл while может использоваться для решения любых задач, два других цикла созданы для удобства пользователя.

Общий формат оператора while:

while (проверка-условия)
оператор;

Проверка условия осуществляется до выполнения оператора или операторов, входящих в тело цикла. В циклах while с несколькими операторами необходимы фигурные скобки:

```
while (проверка-условия)
{
оператор1;
оператор2;
⋮
операторN;
}
```

Выражение, стоящее в круглых скобках вычисляется в цикле while. Если оно истинно, тогда выполняется идущий ниже оператор и выражение вычисляется снова. Если выражение ложно, цикл while заканчивается и программа продолжает свою работу.

Обычно управляющие структуры циклов while используются тогда, когда число повторений цикла - неизвестно.

Цикл for

Цикл for имеется практически во всех языках программирования, включая и Си. Однако, версия этого цикла, используемая в языке Си, отличается большей гибкостью и предоставляет больше возможностей.

Сущностью этого цикла является выполнение набора операторов некоторое определенное число раз, пока некоторая переменная (называемая индексной переменной) не пройдет некоторый промежуток значений.

Общий формат цикла for следующий:

for (инициализация; проверка-условия; коррекция)
оператор;

Внутри круглых скобок стоят три выражения, которые имеют следующее значение:

- инициализация обычно используется для присвоения значения индексной переменной. Она выполняется в начале цикла и больше никогда не повторяется;

- проверка-условия проверяет условие продолжения цикла. Если проверка-условия дает значение "истина", то выполняется один или несколько операторов внутри цикла, если же проверка-условия дало значение "ложь", то операторы внутри цикла и оператор коррекции игнорируются и управление передается оператору, следующему за циклом;

- коррекция обычно модифицирует каким-либо образом индексную переменную. Коррекция выполняется после всех операторов внутри цикла.

Любое из этих трех выражений может быть опущено, но точки с запятой обязательно должны стоять. Если пропущено выражение проверка-условия, считается, что оно имеет значение "истинно", и цикл никогда не заканчивается /бесконечный цикл/.

Цикл for с несколькими операторами внутри выглядит следующим образом:

```
for (инициализация; проверка-условия; коррекция)
{
оператор1;
оператор2;
⋮
оператор N;
}
```

Цикл do...while

Последним мы рассмотрим цикл do...while.

Общий формат цикла do ...while:

```
do
действие;
while (проверка-условия);
```

Основным различием между циклами while и do...while является то, что оператор в цикле do...while выполняется по крайней мере один раз. Цикл do...while выполняется, пока условие истино.

В циклах while с несколькими операторами необходимы фигурные скобки:

```
do
{
действие1;
действие2;
⋮
действие N;
}
while (проверка-условия);
```

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда GNC, Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Написать 3 программы, выполняющие решение задачи 3-мя способами:

а) первая программа решает задачу с применением оператора цикла с параметром (в соответствии с вариантом);

б) вторая программа решает задачу с применением оператора цикла с предусловием (в соответствии с вариантом);

в) третья программа решает задачу с применением оператора цикла с постусловием (в соответствии с вариантом);

Варианты задач:

Вычислить значение функции $y=x^n$, если x=[-F;+F], где F – дата рождения студента (число); n – произвольное натуральное число от 2 до 8.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Получить вариант задания у преподавателя.
3. Разработать схему программы.
4. Составить программы.
5. Отладить программы. Результаты работы программ показать преподавателю.
6. Оформить отчет.
7. Защитить лабораторную работу.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы
5. Текст программы
6. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какой оператор цикла является циклом с параметром?
2. Какой оператор цикла является циклом с предусловием?
3. Какой оператор цикла является циклом с постусловием?
4. Каков формат цикла for?
5. В каком случае выполняется тело цикла for?
6. В каком случае выполняется тело цикла while?
7. В каком случае выполняется тело цикла do- while?

Лабораторная работа №5
Язык С/С++. Массивы и указатели

1. ЦЕЛЬ РАБОТЫ

Целью данной работы является освоение навыков работы с массивами в языке С/С++.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Определение и инициализация массивов

Массивы – набор данных одного типа, собранных под одним именем. Форма объявления одномерного массива:

Тип имя_ массива [размер];

Размер может задаваться константой или константным выражением

Перед использованием массив должен быть объявлен одним из представленных способов.

```
1. #define MAXLINE 10
…
int mas[MAXLINE];
2. int mas[20];
```

В языке Си индексы массива всегда начинаются с нуля, что отражено в циклах for , которые инициализируют и печатают массив. Индекс может быть любым целым выражением.

Для инициализации массива (т.е. присваивания элементам массива начальных значений) в языке Си имеются специальные возможности:

1 способ. В процессе объявление массива указать в фигурных скобках список инициализаторов

```
int MAS[6]={1,2,3,4,5,6};
float MAS[6]={1.1,2.2,3.3,4.4,5.5,6.6};
char MAS[6]={‘a’,’b’,’c’,’d’,’e’,’f’};
```

2 способ. Объявление и инициализация массива без явного указания размера массива

```
int MAS[]={1,2,3,4,5,6};
float MAS[]={1.1,2.2,3.3};
char MAS[]=”adc”;
```

В языке Си предусмотрены прямоугольные многомерные массивы, хотя на практике для массивов размерностью более двух чаще используют массивы указателей.

Многомерные массивы рассмотрим на примере двумерных. Форма объявления двумерного массива: Тип имя_ массива [размер_1] [размер_2];

Размеры задаются константой или константным выражением.

По определению в Си двумерный массив по существу является одномерным массивом, каждый элемент которого является массивом, поэтому индексы записываются как mas[i][j].

В остальном, с двумерными массивами можно обращаться таким же образом, как в других языках. Элементы хранятся по строкам, т.е. при

обращении к элементам в порядке их размещения в памяти быстрее всего изменяется самый правый индекс.

Массив инициализируется с помощью списка начальных значений, заключенных в фигурные скобки; каждая строка двумерного массива инициализируется соответствующим подсписком, заключенным в круглые скобки. Например mas[2][4] = {(0,2,5,4),(1,5,7,3)};

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда GNC, Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Написать программу в соответствии с вариантом задания.

Варианты задания:

Определить и вывести на экран элемент массива:

а) если дата рождения студента – четное число, то минимального;

б) если дата рождения студента – нечетное число, то максимального.

Размерность массива определяется так:

а) если дата рождения студента – четное число, то массив содержит 4 столбца и 5 строк;

б) если дата рождения студента – нечетное число, то массив содержит 5 столбцов и 4 строки.

Элементы массива – натуральные числа, причем:

а) если дата рождения студента – четное число, то в массиве исключены числа 2 и 8.

б) если дата рождения студента – нечетное число, то в массиве исключены числа 3 и 7.

Иные числа в массиве произвольные.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Получить вариант задания у преподавателя.
3. Разработать схему программы.
4. Составить программу.
5. Отладить программу. Результаты работы программы показать преподавателю.
6. Оформить отчет.
7. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы.
2. Цель и задачи.
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы.
5. Текст программы.
6. Результаты и выводы по лабораторной работе.

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В чем особенность объявления и инициализации массива?
2. С какого значения начинаются индексы в массивах?

Лабораторная работа №6
Язык С/С++. Функции

1. ЦЕЛЬ РАБОТЫ

Целью работы является получение навыков написания и использования функций.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Для многократного выполнения одних и те же действий над различными наборами данных необходимо оформить их в виде подпрограммы.

В языке Си подпрограммы называются функциями. Теоретически, каждая функция должна возвращать какое-либо значение. Практически, значения, возвращаемые многими функциями, просто игнорируются. В последних реализациях и определениях языка Си, включая проект стандарта ANSI и Турбо-Си, имеется возможность определения функции типа void, что означает, что эти функции никогда не возвращают никакого значения.

В языке Си функцию можно объявить и определить. При объявлении функции вся программа узнает о ее существовании и все другие функции, включая main, могут к ней обращаться. При определении функции пишется программа, составляющая тело функции.

Константы, типы данных и переменные, объявленные за пределами любой функции, включая main, считаются глобальными, начиная с этого момента. Это значит, что они могут использоваться любой функцией программы, находящейся после их объявления.

Объявление функций

Все функции должны иметь прототипы. Прототипы могут располагаться либо в самой программе на Си, либо в заголовочном файле. Возможно использовать следующий стиль записи прототипов:

возвращаемый-тип имя-функции (тип-аргумента имя-аргумента);

Определение функции

Сама функция содержит в себе некий фрагмент программного кода на Си и обычно следует за описанием функции main(). Функция может иметь вид:

возвращаемый-тип имя-функции (тип-арг1 имя-арг1,…, тип-аргN имя-аргN)

```
{
⋮
/* объявления данных и тело функции) */
⋮
return();
}
```

Функции в программе могут помещаться в любом порядке, они считаются глобальными для всей программы, включая функции, определенные раньше текстуально. Будьте осторожны, используя функцию до того, как она определена или объявлена: когда компилятор встречает неизвестную ему функцию, он присваивает ей тип int. Если вы используете ее

потом и определяете ее, как возвращающую, скажем, char*, вы получите сообщение об ошибке.

Пример. Функция складывает два целых значения и возвращает целое число.

```
# include <stdio.h>
int sum_fun(int x,int y); /* прототип функции (объявление) */
main()
{
int a, b, c;
printf("Введите значения переменных a и b  \n");
scanf("%d%d ", &a, &b);
c=sum_fun(a,b);
printf("Сумма равна %d\n",c); /* сумма */
return (0);
}
int sum_fun(int x,int y) /* определение функции */
{
int z;
z=x+y;
return (z); /* возврат из функции */
}
```

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда GNC, Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Составить программу с использованием функции, разработанной самостоятельно в соответствии с вариантом задания.

Варианты задания:

а) если дата рождения студента – четное число, то функция содержит результат деления даты рождения студента на месяц рождения.

б) если дата рождения студента – нечетное число, то функция содержит результат умножения месяца рождения студента на дату его рождения.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Получить вариант задания у преподавателя.
3. Разработать схему программы.
4. Составить программу.
5. Отладить программу. Результаты работы программы показать преподавателю.
6. Оформить отчет.
7. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы
2. Цель и задачи

3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы
5. Текст программы
6. Результаты и выводы по лабораторной работе

7. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. В каком месте программы объявляются (описываются) функции?
2. Каков формат записи прототипов функции?
3. В каком месте программы определяются функции?
4. Как определяется функция?
5. Каким образом определяется тип функции?
6. Каким образом осуществляется возврат из функции?

Лабораторная работа №7

Создание оконных приложений

1. ЦЕЛЬ РАБОТЫ

Научиться создавать оконные приложения с помощью конструктора форм Windows Forms.

2. ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

В предыдущих лабораторных работах были созданы консольные (неоконные) приложения. Такие программы работают в окнах, напоминающих окна сеансов DOS.

В оконных приложениях используется Windows-интерфейс GUI (Grphical User Interface – графический интерфейс пользователя).

Существует три основных стиля пользовательских интерфейсов:

- однооконный интерфейс (SDI), например, реализованный в WordPad (в WordPad можно открыть только один документ; чтобы открыть другой документ, необходимо открыть первый);
- многооконный интерфейс (MDI), например, реализованный в Microsoft Excel (позволяет отображать несколько документов сразу, при этом каждый документ отображается в отдельном окне);
- интерфейс проводника – это одно окно с двумя панелями или областями; обычно слева представлена иерархия объектов, как в проводнике Microsoft Windows.

Выбор стиля интерфейса зависит от назначения приложения.

Основой графического интерфейса пользователя является форма.

Форма – часть пространства экрана, обычно прямоугольной формы, которую можно использовать для представления сведений пользователю и для получения сведений от него. Форма является основной движущей силой взаимодействия с пользователем. Термин "форма" можно считать синонимом окна – окна приложения, диалогового окна и т.д.

Значительная часть пользовательского интерфейса приложений реализуется именно с применением диалоговых окон. Это окна, предназначенные для открытия, сохранения, печати и закрытия документов, окна отображения и настройки всевозможных параметров и т.д.

Диалоговые окна принято делить на модальные и немодальные окна. Когда приложение открывает на экране модальное окно, его работа будет приостановлена до тех пор, пока пользователь не закроет это окно. Что же касается немодальных окон, то они работают одновременно с главным окном открывшего их приложения. Например, диалоговое окно сохранения документа– модальное, пока пользователь не ответит на поставленный вопрос работа с документом Microsofr Office Word будет невозможной.

Форму можно полностью создать с помощью редактора кода. Однако, для создания и изменения форм проще использовать конструктор Windows Forms. Для создания новой формы с помощью конструктора форм необходимо выполнить команду File→New→Project→Windows Forms Application. На форме размещаются элементы управления.

Элемент управления (или управляющий элемент, программный элемент, компонент Windows-форм, или контрол) – это объект на форме, который придает форме новые функциональные возможности и формирует пользовательский интерфейс. Например, кнопки Да, Нет, Отмена и текст сообщения являются элементами управления.

Элементы управления расположены в окне Toolbox. Вызов Toolbox может быть выполнен несколькими способами: View→Toolbox, кнопка на панели инструментов или с помощью пункта Toolbox бокового меню.

Элементы управления в окне Toolbox сгруппированы по нескольким вкладкам: All Windows Forms, Common Controls и др.

Рассмотрим назначение некоторых элементов управления.

Button (кнопка) служит для выполнения действия с помощью мыши. При щелчке кнопки вызывается обработчик события Click. В обработчик события Click помещается код, отвечающий за выполнение нужного действия.

CheckBox описывает селекторную кнопку со свободной фиксацией (предоставляет выбор нескольких элементов).

ComboBox используется для выбора одного значения из нескольких возможных.

Label (метка) используется для отображения текста или рисунков, которые не могут редактироваться пользователями. Они используются для определения объектов в форме.

LinkLabel предназначено для добавления в форму ссылок на ресурсы Интернета, такие как адреса Web-сайтов и серверов FTP, адреса электронной почты и пр.

PictureBox используется для привязки к форме файла графического изображения (операция масштабирования изображения недоступна)

RadioButton описывает селекторную кнопку с зависимой фиксацией (если одна кнопка включена, другие обязательно выключены). Похож на CheckBox, за исключением того, что, элементы выбора взаимоисключающие.

TextBox (текстовое поле) используется для получения данных, вводимых пользователем, или для отображения текста.

MaskedTextBox (текстовое поле по маске) – позволяет задать маску для пользовательского ввода. Например, пользователь должен вводить номер телефона с кодом области, тогда установив свойство этого контрола Mask в (999)000-0000 мы позволим вводить данные в таком формате.

MenuStrip – позволяет добавить в стандартное меню иконку.

Для размещения нужного элемента управления достаточно просто щелкнуть на нем в окне Toolbox или, ухватив, перетащить его на форму. Размещение элементов на форму происходит в окне Конструктора формы.

Элементы управления в Visual C++ на форме легко выровнять и упорядочить. Например, при размещении элемента управления к краю формы можно увидеть ограничительные линии, отмечающие собой рекомендуемое расстояние при размещении элементов на форме. То же самое касается и взаимодействия элементов управления между собой. Также можно

воспользоваться панелью Layout, которая становится активной при выбранных элементах.

Свойства (Properties) формы и размещенных на ней компонентов отображаются при нажатии на кнопку Properties Window на панели инструментов или выбора пункта Properties из контекстного меню, когда соответствующий объект выделен.

Свойства формы и элементов управления сгруппированы по вкладкам.

Двойной щелчок на элементе управления перемещает нас в окно кода, где выполняется работа с кодом программы. Т.о., при создании проектов Windows Forms происходит работа с двумя окнами – окном конструктора форм (Design) и окном кода.

3. ОБОРУДОВАНИЕ

ПЭВМ IBM PC, SVGA монитор с разрешением не менее 800*600 пикселей; клавиатура; мышь. Среда GNC, Visual Studio C++.

4. ЗАДАНИЕ НА РАБОТУ

Составить программу в соответствии с вариантом задания.

Варианты задания:

а) Если дата рождения студента заключена в интервал от 1 до 10, то программа должна иметь оконный интерфейс и должна определять количество элементов в произвольном двумерном массиве. Размер массива – nxn, где n – натуральное число в интервале от 5 до 10.

б) Если дата рождения студента заключена в интервал от 11 до 20, то программа должна иметь оконный интерфейс и должна определять сумму элементов в произвольном двумерном массиве. Размер массива – nxn, где n – натуральное число в интервале от 5 до 10.

в) Если дата рождения студента заключена в интервал от 21 до 31, то программа должна иметь оконный интерфейс и должна определять произведение отрицательных элементов в двумерном массиве. Размер массива – nxn, где n – натуральное число в интервале от 5 до 10.

5. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические положения.
2. Получить вариант задания у преподавателя.
3. Разработать схему программы.
4. Составить программу.
5. Отладить программу. Результаты работы программы показать преподавателю.
6. Оформить отчет.
7. Защитить лабораторную работу перед преподавателем.

6. СОДЕРЖАНИЕ ОТЧЕТА

1. Номер и название лабораторной работы.
2. Цель и задачи.
3. Задание на работу. Описание задания в соответствии с вариантом.
4. Схема программы.
5. Текст программы.
6. Результаты и выводы по лабораторной работе.