ФГБОУВПО «Воронежский государственный технический университет»
кафедра компьютерных интеллектуальных технологий проектирования

264-2011

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к практическим работам по дисциплине для студентов направления 230100.64 «Информатика и вычислительная техника» профиля «Системы автоматизированного проектирования в машиностроении» очной формы обучения
Часть 1

Воронеж 2011

Составители: канд. техн. наук А.Н. Юров,
канд. техн. наук М.В. Паринов,
д-р. техн. наук М.И. Чижов,
ст. преп. В.А. Рыжков

УДК 004.9

Методические указания к практическим работам по дисциплине для студентов направления 230100.64 «Информатика и вычислительная техника» профиля «Системы автоматизированного проектирования в машиностроении» очной формы обучения. Ч.1 / ФГБОУ ВПО «Воронежский государственный технический университет»; сост. А.Н. Юров, М.В. Паринов, М.И. Чижов, В.А. Рыжков. Воронеж, 2011. 39 с.

Методические указания содержат перечень базовых разделов с примерами решения задач на языке С++ по дисциплине «Программирование на языках высокого уровня».

Предназначены для студентов 1 курса.

Методические указания подготовлены в электронном виде в текстовом редакторе Microsoft Word 2007.

Ил. 4. Библиогр.: 10 назв.

Рецензент д-р техн. наук, проф. В.Н. Старов

Ответственный за выпуск зав. кафедрой д-р техн. наук, проф. М.И. Чижов

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ВВЕДЕНИЕ

Эффективность процесса обучения специалистов по курсу ”Программирование на языках высокого уровня” составляют, в основном, практические занятия и лабораторные работы, которые формируют общее представление о предмете и позволяют самостоятельно понять принципы решения вычислительных задач. Регулярная практическая работа в написании программ позволяет усвоить основы теоретического материала. Она способствует развитию творческого мышления, умению самостоятельно применять теоретические знания на практике, создавать программные средства с максимальными возможностями по производительности, а также элегантности решений.

В предложенных методических указаниях по практической работе рассматриваются следующие цели и задачи:

- систематизировать, закреплять теоретические знания на практике в написании программных средств;
- выработать навыки к интеллектуальному творчеству;
- оценить общий уровень обучаемого студента;
- сформировать навыки оформления программных проектов согласно рабочей программе и требованиям кафедры;
- подготовить студента к последующей работе на старших курсах обучения.

В рассматриваемых методических указаниях приводятся некоторые теоретические сведения из ряда разделов изучаемого курса по программированию, приведены рекомендации к выполнению расчетных задач, а также приводятся задания для самостоятельного выполнения. Требования составлены в соответствии с государственным стандартом и учебным планом выпускающей кафедры.

1. РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ НА ЯЗЫКЕ С++

1.1. Общие сведения о языке разработки и интегрированных средствах создания программных проектов

Язык С++ является универсальным средством разработки проектов для создания приложений под различные операционные системы персональных ЭВМ, мобильных устройств и бытовой техники. На указанном языке создаются программы, сервисные утилиты, компоненты, которые обслуживают всевозможные сервисы и оборудование вычислительных устройств. Более того, язык, благодаря своей популярности, явился основой для целого ряда иных языков, успешно решающих задачи программного проектирования приложений. Разработанный в 1978 году, язык обладал производительностью, сравнимой с машинными языками и ассемблерами по некоторому ряду задач. Простые и понятные конструкции языка включали в себя новый, объектно-ориентированный подход, позволяя разделить область памяти ЭВМ рациональным способом, а также защитить данные от возможных сбоев посредством распределения ресурсов системы. Язык активно развивается и по настоящий момент. Комитет по стандартизации языков программирования несколько раз утверждал спецификацию языка С++, при этом постоянно вносились изменения в действующий формат и концепцию средств реализации системы программирования на С++.

В настоящий момент имеется целый ряд систем, позволяющих вести разработку проектов на С++. Комплексные системы разработки называются интегрированными средами. Существуют коммерческие программные продукты, включающие визуальные средства разработки, профессиональные элементы отладки, интеллектуальные редакторы с возможность навигации в

проекте, а также встроенную систему помощи по структуре языка. К таким средствам разработки относятся: Visual Studio 2010 от Microsoft, RAD Studio XE, включающая Builder C++ от CodeGear, XCode от Apple и многие другие. Существуют и свободно распространяемые интегрированные среды проектирования приложений, такие как: Code::Blocks Studio, CodeLite, QTCreater, wxDev C++, Eclipse и т.д.

1.2. Разработка консольных приложений на C++

Пример программы, использующий функции ввода-вывода с описанием основных компонентов и блоков. В программе рассматриваются основные математические операции: “+,-,/,*”. Далее представлена реализация проекта:

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a,b;
    //Приглашение к вводу двух чисел
    cout<<"Enter a two numbers"<<endl;
    cin>>a>>b;
    //Реализация математических действий
    cout<<"Add operation:"<<a+b<<endl;
    cout<<"Sub operation:"<<a-b<<endl;
    cout<<"Mul operation:"<<a*b<<endl;
    cout<<"Div operation"<<a/b<<endl;
    system("PAUSE");
   return 0;
}
```

Как видно из предложенного листинга, текст программы включает менее 20 строк, причем при наборе текста программы можно подготовить всю реализацию из функций в одной строке. Текстовые редакторы допускают такую возможность. Однако, с точки зрения читаемости текста программы, такие действия не рекомендуются. Более того,

существует ряд соглашений, в которых регламентируются правила набора команд, стиль, способ объявления переменных и прочие действия по ведению проектов. Для коммерческих организаций и групп разработчиков такие мероприятия позволяют быстрее изучать имеющиеся и разрабатывать новые проекты, получая при этом некоторые стандарты в данной области.

Вся реализация проекта помещена в функцию main()-ту базовую основу, на которой построена концепция консольных приложений. Функция main присутствует с некоторыми изменениями и под Windows, при использовании графических, а также промежуточных технологий, на которых строятся приложения. В разрабатываемом приложении функция main может быть только одна! Текст программы, который будет выполняться процессором в ходе работы приложения, заключается в фигурные скобки {}. Все, что напечатано за пределом скобок, к исполняемому тексту основной функции относиться не будет. Исключения составляют директивы препроцессора, настройки среды, вынесенные за пределы заголовки других функций с их реализаций. В рассматриваемом варианте следующий текст представлен директивами предпроцессора, которые подключают заголовки библиотеки ввода-вывода (`#include <iostream>`) и стандартную библиотеку (`#include <cstdlib>`).Директивой using namespace std; определяется стандартное пространство имен, что делает возможным использование функции-ввода вывода. Функция main() в круглых скобках может содержать параметры. Иногда при запуске программы бывает полезно передать ей какую-либо информацию. Обычно такая информация передается функции main() с помощью аргументов командной строки. Аргумент командной строки — это информация, которая вводится в командной строке операционной системы вслед за именем программы. Чтобы принять аргументы командной строки, используются два специальных встроенных аргумента: argc и argv. Параметр argc

содержит количество аргументов в командной строке и является целым числом, причем он всегда не меньше 1, потому что первым аргументом считается имя программы. А параметр argv является указателем на массив указателей на строки. В этом массиве каждый элемент указывает на какой-либо аргумент командной строки. Все аргументы командной строки являются строковыми, поэтому преобразование каких бы то ни было чисел в нужный двоичный формат должно быть предусмотрено в программе при ее разработке.

Зарезервированное ключевое слово int перед функцией main указывает на тип возвращаемого значения. Программы обычно передают операционной системе некоторую информацию после того, когда завершена работа приложения. При разработке программ можно вернуть определенное значение, которое, впоследствии проанализировав, принять решении о работоспособности проекта в целом. Для возвращения некоторого числового значению используется функция return n, где n-произвольное число в заданном диапазоне типа int. Тип int позволяет определить работу с целочисленными значениями элементов программы. В проекте определены два числа: a и b целого типа. Строка, содержащая символы ”//”, является комментарием к тексту программы, а последующие символы в строке не рассматриваются компилятором. Ввод переменных осуществляется функцией cin с комбинацией символов ”>>”. Переменные, подлежащие вводу с клавиатуры, достаточно в тексте просто перечислить в строке, используя множественную аббревиатуру из указанных знаков ”>>”. Для вывода результатов в консольное окно в C++ используется перегруженная функция cout с набором знаков ”<<”. Понятие перегруженности будет разбираться в последующих работах, сейчас же необходимо отметить следующее - в функции реализована возможность выводить числовую и символьную информацию, поддерживая разные типы данных. Строка system(“PAUSE”) позволяет организовать паузу в работе

приложения, используя системную команду операционной системы. Прототип функции system взят из стандартной библиотеки, заголовок которой подключен по директиве предпроцессора #include <cstdlib>.

Реализация и выполнение проекта выполнялась в кроссплатформенной среде IDE QTCreater, представленной на рисунке 1.

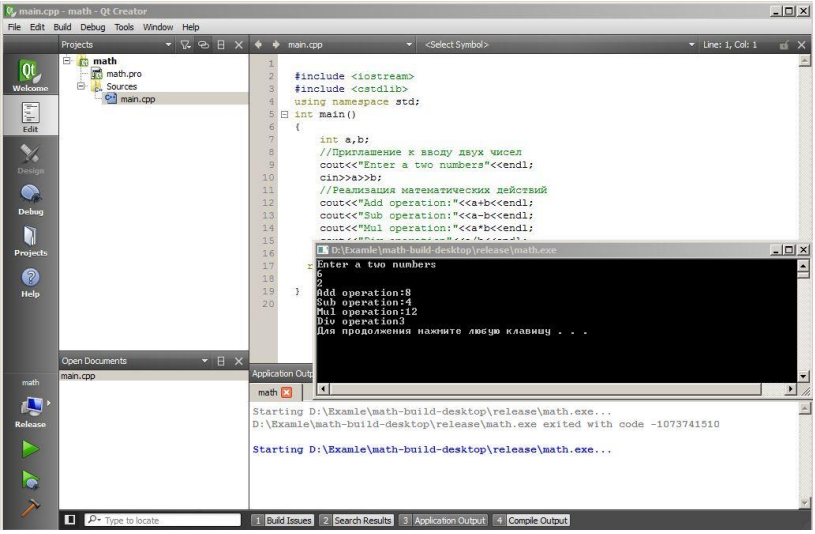

Рис. 1. Среда разработки приложений QTCreator

Среда разработки QT Creator представляет собой комплексное средство для создания и отладки приложений, разрабатываемые под ОС, в том числе и мобильные. На рисунке 2 показаны все типы проектов для решения практических задач: подготовка пользовательского графического интерфейса средствами QT, использование интерфейса QT для мобильных систем и разработка консольных приложений. В методических указаниях рассматриваются и создаются консольные проекты, которые легко могут быть перенесены и на другие IDE решения Windows, Linux, UNIX, MacOS и другие. Выбрав соответствующий раздел с проектом, пользователю

предоставляется некоторый шаблон, который доступен для редактирования и дополнения. Минимальный консольный проект в QT состоит из двух файлов: с расширением .pro и main.cpp.

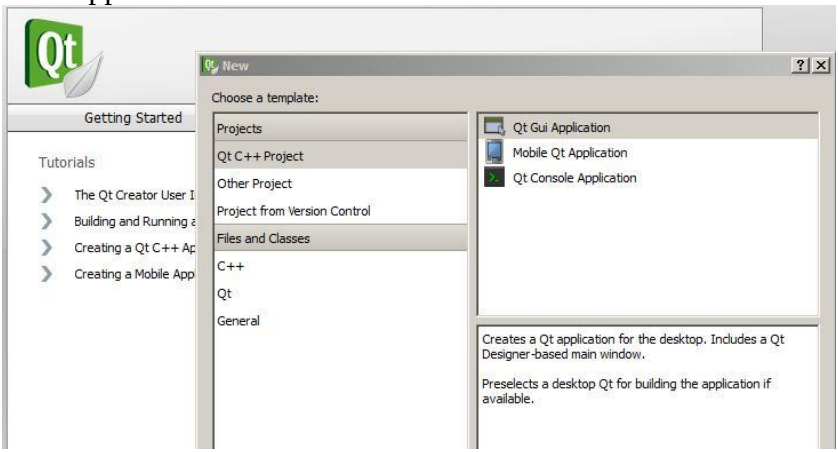

Рис. 2. Выбор типа проектного решения

Рассмотрим проект с расширением .pro. Проектный файл показан на рисунке 3.

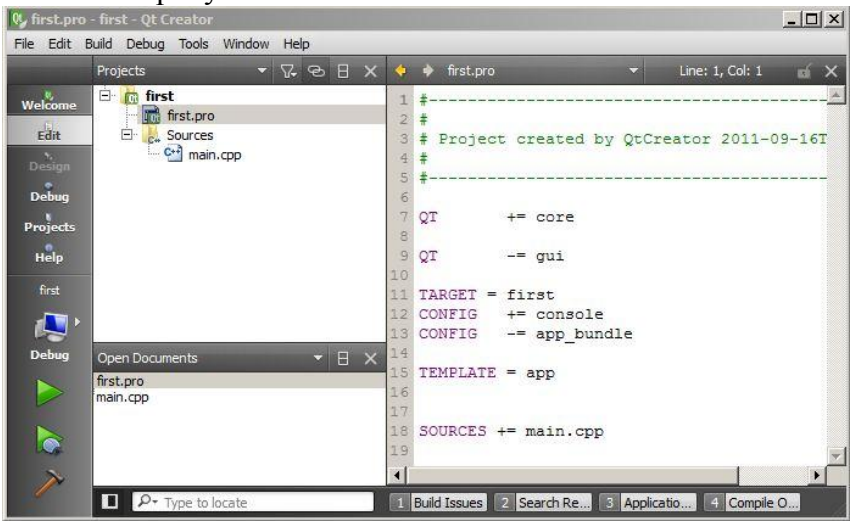

Рис. 3. Состав проектного файла для консольного приложения

```
QT       += core
QT       -= gui
TARGET = first
CONFIG   += console
CONFIG   -= app_bundle
TEMPLATE = app
```
SOURCES += main.cpp

Первая строка в файле означает, что в основе приложения будет использоваться ядро QT, которое представляет собой некоторую программную ”обертку” для программного средства. Появляются возможности использовать классы QT с реализацией готовых методов с интерфейсом, подключением сетевых ресурсов и компонентов, управлением ряда устройств и т.д. Вторая строчка отменяет использование графического интерфейса для консоли. Для других типов проектов такая опция будет включена. За включение будет отвечать комбинация знаков ”+=”. Строка, начинающаяся со слова ”TARGET”, указывает на имя исполняемого файла, который будет создан системой после сборки всего проекта. Имя проектного файла и имя исполняемого могут отличаться друг от друга. В строке, отвечающей за конфигурацию проекта (CONFIG), указан тип для проекта - в данном случае консольный, а шаблон представляет собой приложение под ОС. Последняя строка отвечает за единственный имеющийся в проекте файл main.cpp, в котором можно вести разработку программы. Конечно же большинство проектов не ограничиваются одним файлом, поэтому в проект имеется возможность добавить свои файлы, содержащие не только текст, но и ресурсы для будущей системы в целом.

Файл main.cpp, предлагаемый средой QT, выглядит следующим образом.

```
#include <QtCore/QCoreApplication>
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    return a.exec();
```

```
}
```

Из текста видно, что при проектировании выполняется некоторая эмуляция в адресном пространстве QT, так как весь проект строится на основании ядра системы в целом. Однако при разработке консольных приложений имеется возможность отказаться от подключаемых заголовочных файлов QT, что и было реализовано в первой проекте.

Подключение заголовочных файлов значительно расширяют возможности системы. Например, при подключении математической библиотеки C++ появляется возможность использовать ряд функций, которые используются для решения расчетных задач. Заголовочный файл подключается согласно директиве #include <cmath>. Для совместимости с интерфейсом C оставлена возможность по использованию заголовка <math.h>. Далее приводится некоторый список с описанием математических инструкций.

pow(x,y)-функция, которая позволяет возвести число x, d степень y. Нетрудно заметить, что функция будет универсальна для вычисления корня из x, если значение y будет следующим: 0<y<1. Имеются подобные функции powf() и powl(), которые работают с иными типами данных и добавлены в версии C99. Далее приводится пример работы указанной математической функции.

```
//Программа выводит первые 3 степени числа 10
#include <cmath>
#include <stdio.h>
#include <cstdlib>
using namespace std;
int main()
 {
      double x = 10.0, y = 0.0;
        printf("%f\n", pow(x, y));
        y++;
        printf("%f\n", pow(x, y));
        y++;
        printf("%f\n", pow(x, y));
        y++;
```

```
        printf("%f\n", pow(x, y));
        y++;
      system("PAUSE");
    return 0;
}
```

В программе использован вывод значений на экран по спецификации языка С через printf. Такая конструкция допустима наряду с выводом посредством cout. Операция y++ означает, что необходимо увеличить значение переменной на 1.

Рассмотрим другие функции математической библиотеки.

exp()- функция возвращает значение экспоненты от аргумента (число *e*, возведенное в степень, которая равна значению аргумента *arg*).

log() – функция возвращает значение натурального логарифма от указанного аргумента. Если значение аргумента отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений (ошибка из-за нарушения области определения). Если же значение равно нулю, возможна ошибка из-за выхода за пределы диапазона представимых значений.

sqrt() – функция возвращает значение квадратного корня от аргумента. Если значение аргумента отрицательно, возникает ошибка из-за выхода за пределы области допустимых значений. Имеются подобные функции, которые отличаются типом входящего параметра: sqrtf() и sqrtl() Добавлены в версии C99.

rint() – функция возвращает значение аргумента, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой.

round() - функция возвращает значение аргумента, округленное до ближайшего целого. Однако возвращаемое число представлено в формате с плавающей точкой. Значения, отстоящие от большего и меньшего целого на одинаковую

величину (например, число 3.5), округляются в сторону большего целого.

remquo() - функция возвращает остаток от деления значений аргументов *a/b*. При этом целое, адресуемое параметром *quo*, будет содержать частное.

fma() - функция возвращает значение выражения *a*b+c*. Округление выполняется только один раз, после завершения всей операции.

`sin()` – функция возвращает значение синуса от указанного значения аргумента. Значение аргумента должно быть задано в радианах.

`asin()` – функция возвращает значение арксинуса от аргумента. Значение аргумента должно находиться в диапазоне от -1 до 1; в противном случае возникает ошибка из-за выхода за пределы области допустимых значений.

Аналогично работают функции математической библиотеки, связанные с вычислением косинуса (арккосинуса) и тангенса (арктангенса).

Задачи на самостоятельное решение

1.Вычислить значение функции y=sin(x)+3e^x при x=2,5; -4,4;0,001.

2. Определить значение выражения y=15/log(x)+arctan(z/x)*1/5 при z и x, определяемых вводом с клавиатуры.

3. Используя функции математической библиотеки вычислить значения корня от 15 до 20 для следующего выражения: cos(x)+$x^{4.5}$. Значение x определить вводом с клавиатуры.

2. ТИПЫ ДАННЫХ

Все данные, используемые в программе, можно разделить на две группы: константы и переменные. К первой группе относятся данные, не изменяющие своего значения в ходе

выполнения программы, данные второй группы могут изменять свое значение.

Типы и размерность данных

Тип данных	Размер памяти, бит	Диапазон значений
char (символьный)	8	от–128 до 127
signed char (знаковый символьный)	8	от–128 до 127
unsigned char (беззнаковый символьный)	8	от 0 до 255
short int (короткое целое)	16	от–32768 до 32767
unsigned int (беззнаковое целое)	16	от 0 до 65535 (16-битная платформа) от 0 до 4294967295 (32-битная платформа)
int (целое)	16 32	от–32768 до 32767 (16-битная платформа) от –2147483648 до 2147483647 (32-битная платформа)
long (длинное целое)	32	от–2147483648 до 2147483647
long long int (C99)	64	от–(2^{63}–1) до 2^{63}–1
unsigned long long int (C99)	64	от 0 до 2^{64}–1
float (вещественное)	32	от 3.4E–38 до 3.4E38
double (двойное вещественное)	64	от 1.7E–308 до 1.7E308
long double (длинное вещественное)	80	от 3.4E–4932 до 3.4E4932
_Bool (C99)	8	true(1), false(0)
bool (C++)	8	true(1), false(0)

Числовые типы данных языка C++ представлены в таблице 2.1.

Как константы, так и переменные могут быть различных типов, которые определяют их структуру, набор допустимых значений, правила использования и способ представления в ЭВМ.

В 32-разрядной ОС Windows типint занимает в памяти 32 бита, и диапазон допустимых значений для знакового int в этом случае от минус 2147483648 до 2147483647. Такое различие в размере памяти, выделяемой под переменную типа int, объясняется тем, что тип int – машинно-зависимый, и для него выделяется одно машинное слово, длина которого в 16-разрядных процессорах – 16 бит, в 32-разрядных – 32 бита. Рассмотрим примеры на использование применяемых типов данных в консольном приложении в QT.

Выполнить перевод значений температур в градусах по Фаренгейту в градусы по Цельсию.

```
#include <iostream>
#include "conio.h"
#include <clocale>
using namespace std;
int main()
{
//Локализация под национальный формат вывода символов
setlocale(LC_ALL,"Russian");
cout<<"Программа для перевода температуры в градусах
по Фаренгейту в градусы Цельсия"<<endl;
cout<<"Введите температуру в градусах по
Фаренгейту:"<<endl;
float foreng;
cin>>foreng;
float celsi;
celsi=((foreng-32)*5)/9;
cout<<"Температура в градусах Цельсия: "<<celsi;
getch();
return 0;
}
```

В программе используется вывод кириллических символов при отображении информационных сообщений. Такая возможность появляется после выполнения функции setlocale() с указанием второго параметра того или иного языка -в нашем случае ”Russian”. Для того чтобы действия распространились на весть проект используется первый параметр с описанием LC_ALL. Прототип команды setlocale() указан в подключаемом файле <clocale>.

Функция getch() предусматривает в программе задержку для просмотра результата. Указанная функция ожидает нажатия любой клавиши со стороны оператора и возвращает соответствующий символ для последующих манипуляций с ним.

Следующая программа производит посредством использования sizeof() вычисление размерности применяемых типов в языке C++.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    cout<<"int:"<<sizeof(int)<<"-byte of
information"<<endl;
    cout<<"short:"<<sizeof(short)<<"-byte of
information"<<endl;
    cout<<"long long:"<<sizeof(long long)<<"-byte of
information"<<endl;
    cout<<"char:"<<sizeof(char)<<"-byte of
information"<<endl;
    cout<<"bool:"<<sizeof(bool)<<"-byte of
information"<<endl;
    cout<<"unsigned:"<<sizeof(unsigned)<<"-byte of
information"<<endl;
    cout<<"float:"<<sizeof(float)<<"-byte of
information"<<endl;
    system("PAUSE");
    return 0;
}
```

Оператор `sizeof` подсчитывает размер любой переменной или любого типа и может быть полезен, если в программах требуется свести к минимуму машинно-зависимый код. Этот оператор особенно полезен там, где приходится иметь дело со структурами или объединениями. По строке кода cout<<sizeof(long)<<endl; на экране будет выведено значение числа, равное 4, а комбинация “long long” приведет к отображению 8.

Задачи на самостоятельное решение

1.Написать программу, которая вводит число из пяти цифр, разделяет число на отдельные цифры и печатает их отдельно друг от друга с тремя пробелами между ними. Например, если пользователь вводит в программу 42339, то должно быть напечатано 4 2 3 3 9.

2. Составить программу, в которой вводится четырехзначное число (например, 5936), а вывод чисел на экран осуществляется в обратном порядке (6395).

3. Для значения 567,23 (возможен ввод отличного значения с клавиатуры) получить квадраты значений каждого из разрядов предложенного числа.

3. ПЕРЕМЕННЫЕ И КОНСТАНТЫ

3.1. Локальные и глобальные переменные

Большинство программ в процессе работы манипулирует данными, которые определяются разработчиком в процессе создания программного средства. Дальнейшие изменения производят пользователи, когда эксплуатируют программу. С точки зрения внутреннего представления архитектуры ЭВМ переменная — это именованный участок памяти, в котором хранится значение, которое может быть изменено программой.

Все переменные перед их использованием должны быть объявлены.

Для того чтобы определить изменяемую часть данных минимального размера, необходимо создать переменную. Для этого достаточно в любом месте листинга в фигурных скобках определиться с именем и типом данных будущей переменной. Пример создания переменной вещественного типа:

```
int main()
{
        float my_value;
        my_value=3.3;
        return 0;
}
```

В приведенном примере переменная вначале объявляется, а в следующей строке кода ей присваивается значение. Допускается объявлять тип переменной с присвоением начального значения, например так: int temp=1.

Переменные могут быть объявлены в любой части программы, однако при объектно-ориентированном подходе ряд переменных выносится в отдельный блок, как и функции-методы класса. Рекомендуется при разработке переменные выносить в начало программы и присваивать информативно-понятные имена для них, а также комментировать назначение той или иной переменной. Имена переменных, имеющих одни и те же символы, но набранные прописными или заглавными символами, различны друг от друга. Т.е. double Pie,PIE несмотря на то, что объявлены перечислением, отличаются регистром символов и представляют собой 2 обособленные переменные.

Глобальные переменные представляют собой такие элементы, которые сохраняют свое действие на всем протяжении программы. Действие локальной переменной распространяется только на тот блок программы, где было сделано объявление. Следующий пример демонстрирует возможности такого объявления и завершается ошибкой, если

производится вывод значений за пределом действия блока объявления.

```
//Объявление в программе переменных
#include "iostream"
#include "cstdlib"
using namespace std;
int main()
{
        int global=3;
        {
            int local=4;
        }
        cout<<global<<endl;
        cout<<local<<endl;
}
```

В примере объявлены 2 переменные, но так как local объявлена во внутреннем блоке, действие этой переменной завершается, как только будет закрыта скобка-ограничитель блока. Последующий вывод local приведет к ошибке, что демонстрируется компилятором среды на рисунке 4.

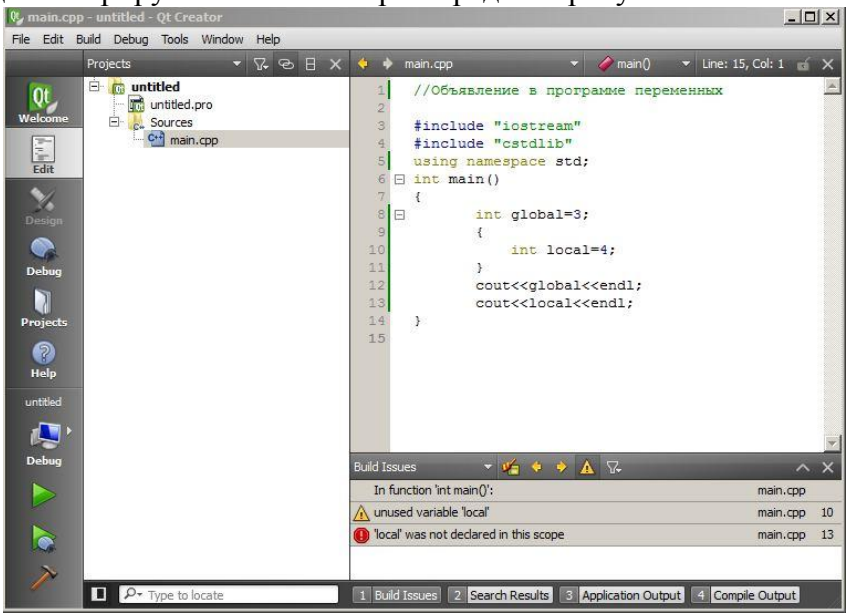

Рис. 4. Ошибка сборки проекта при выводе сообщения

3.2. Константы и способы их задания

Под константой понимают некоторую величину, которая не изменяется на протяжении работы программы. Способы задания констант аналогичны определению переменных с той лишь разницей, что перед типом данных необходимо указать ключевое слово const. Константа может относиться к любому базовому типу. Способ представления константы определяется ее типом. Константы также называются литералами.

```
const float m_steel=7.8;
const float m_water=1.0;
const int m_temp=-15;
```

Символьные константы заключаются в одинарные кавычки. Например, 'a' и '%' — это символьные константы. В языке C++ определены многобайтовые (состоящие из одного или более байт) и широкие (обычно длиной 16 бит) символы. Они используются для представления символов языков, имеющих в своем алфавите много букв. Многобайтовый символ записывается в одинарных кавычках, например, 'xy', а широкий — с предшествующим символом L, например:

```
wchar_t wc;
wc = L'A';
```

Здесь переменной wc присвоено значение константы A, рассматриваемой как широкий символ. Тип расширенного символа wchar_t определен в заголовочном файле <cstddef>, этот тип не является встроенным.

Используя суффикс в конце определения числа, можно явно указать тип числовой константы. Если после числа в плавающем формате стоит суффикс F, то считается, что константа имеет тип float, а если L, то long double. Для целых типов суффикс U означает unsigned, а L — long. Тип суффикса не зависит от регистра, например, как F, так и f определяют константы типа float. Приведем пример программы, в которой рассмотрен ряд констант.

```
//Объявление в программе констант
#include "iostream"
#include "cstdlib"
#include <cstddef>
using namespace std;
int main()
{
  const int value_int=35000L;
  const unsigned int value_u_int=10000U;
  const float value_float=4.34e-4f;
  const long double value_double=1001.2L;
}
```

В ряде случаев удобнее использовать не десятичную, а восьмеричную или шестнадцатеричную систему. Позиционную систему счисления (в этой системе порядок записи чисел является определяющим в форме представления числа) с основанием 8 называют восьмеричной. В ней используются цифры от 0 до 7. Число 10 в восьмеричной системе представляет то же число, что и 8 в десятичной. Позиционная система счисления с основанием 16 называется шестнадцатеричной. В ней используются 16 символов: цифры от 0 до 9 и символы от A до F, обозначающие цифры от 10 до 15. Например, запись 10 в шестнадцатеричной системе обозначает то же число, что и 16 в десятичной системе. Эти системы счисления используются довольно часто, поэтому в языке C++ целые константы можно определять не только в десятичной, но и в восьмеричной и шестнадцатеричной системах счисления. Шестнадцатеричная константа начинается с 0x, а восьмеричная — с 0. В программе показаны способы задания чисел в двух системах.

```
//Шестнадцатеричная и восьмеричная
//форма записи числе в программе
#include "iostream"
#include "cstdlib"
#include <cstddef>
#include <windows.h>
using namespace std;
int main()
```

```
{
   //Русификация
   SetConsoleCP(1251);
   SetConsoleOutputCP(1251);
   //131 в десятичной системе
   const int hex = 0x83;
   //21 в десятичной системе
   const int oct = 025;
   cout<<"Шестнадцатиричное представление"<<hex<<endl;
   cout<<"Восьмиричное представление"<<oct<<endl;
   system("PAUSE");
   return 0;
}
```

Задачи на самостоятельное решение

1. Даны два целых числа a и b. Если a делится на b или b делится на a, то вывести на экран 1, иначе - любое другое число. Условные операторы и операторы цикла не использовать.

2. С начала суток часовая стрелка повернулась на некоторый угол. Угол представлен целым числом от 0 до 360 градусов. Определить число полных часов и число полных минут, прошедших с начала суток.

3. В программе организовать ввод произвольного числа, содержащего три или более разряда и провести суммирование первого и последнего разрядов с выводом результата на экран.

4. ОПЕРАЦИИ И СИНТАКСИС

В языке C++ используется значительное количество операция, особенно если сравнивать его с другими языками и средствами их реализации. Один и тот же символ, в зависимости от ситуации, может быть использован в отличных друг от друга задачах.

Существует четыре основных группы выполняемых действий и операций: арифметические, операции сравнения,

логические и поразрядные. Кроме них, есть также некоторые специальные операторы, например, оператор присваивания.

4.1. Арифметические операции

Базовые операции, которые обычно используются в записи выражений, имеют то же назначение, что и в других языках программирования. К ним относятся:

”-”-операция вычитания (или унарный минус);

”+”-операция сложения;

“*”-операция умножения;

“/”- операция деления;

“%” –остаток от деления. Пример 5%2=1;

“++”-увеличение числа на 1 (инкремент);

“--”-уменьшение числа на 1 (декремент);

Операции, применяемые в выражениях, имеют тот же приоритет что и арифметические операции в математике. Операции по увеличению и уменьшению числа на 1 являются наиболее употребляемыми в программировании, поэтому такое решение виде комбинации знаков ”--” и “++” часто входит в состав конструкций повторения. Иными словами операции можно записать так:

```
x = x+1;
```

и так:

```
++x;
```

Аналогично форма представления выражения по уменьшению значения на 1:

```
x = x-1;
```

и при использовании операции декремента:

```
x--;
```

Как инкремент, так и декремент могут предшествовать операнду (префиксная форма) или следовать за ним (постфиксная форма). Следующие строки демонстрируют такие возможности:

```
x = x+1;
```

можно записать следующим образом:

++x;

или использовать следующую форму записи:

x++;

Однако префиксная и постфиксная формы отличаются при использовании их в выражениях. Если оператор инкремента или декремента предшествует операнду, то сама операция выполняется до использования результата в выражении. Если же оператор следует за операндом, то в выражении значение операнда используется до выполнения операции инкремента или декремента. Например,

x = 10;

y = ++x;

присваивает y значение 11. Однако если написать

x = 10;

y = x++;

то переменной y будет присвоено значение 10. В обоих случаях x присвоено значение 11, разница только в том, когда именно это случилось, до или после присваивания значения переменной y.

Выражение с операцией ”x++” и “x--” обрабатывается процессором в выполняемом коде эффективнее, чем аналогичное выражение ”x=x+1” и ”x=x-1”, поэтому везде, где это возможно, рекомендуется использовать инкремент и декремент.

Приоритет выполнения арифметических операторов следующий: инкремент и декремент, унарный минус, операции умножения и деления, сложение и вычитание.

Операции с одинаковым приоритетом выполняются слева направо. Используя круглые скобки, можно изменить порядок вычислений. В языке C++ круглые скобки интерпретируются компилятором так же, как и в любом другом языке программирования – скобки назначают операции (или последовательности операций) наивысший приоритет.

Операция присваивания уже рассматривалась при разработке программ в математических и прочих выражениях, а также при определении начальных значений переменных и создания констант. Объявление переменной с присваиванием выглядит следующим образом: int dig=15. В языке C++ допускается использование комбинаций арифметических операций с присваиванием следующего рода: “/=”,”+=”,”*=”,”-=” и т.д., так называемое составное присваивание. В следующей программе показано применение составного присваивания

```
//Сокращения, применяемые в математических операциях
#include "iostream"
#include "cstdlib"
#include <cmath>
using namespace std;
int main()
{
   double y=3;
   y++;
   y/=0.8;
   y*=8;
   y+=y/3;
   //Результатом выполнения будет число, равное 52,333
   cout<<"Importance y is:"<<--y<<endl;
   system("PAUSE");
   return 0;
}
```

Если ряд переменных требуется проинициализировать одним значением, то можно использовать в программе следующую конструкцию:

```
int a,b,c;
a=b=c=0;
```

4.2. Операции сравнения и логические операции

Операции сравнения - это операции, в которых значения двух переменных сравниваются между собой. Логические же

операции реализуют средствами языка операции формальной логики. Между логическими операциями и операциями сравнения существует тесная связь: результаты операций сравнения часто являются операндами логических операций.

В операциях сравнения и логических операциях в качестве операндов и результатов операций используются значения ИСТИНА (true) и ЛОЖЬ (false). В языке С++ значение ИСТИНА представляется любым числом, отличным от нуля. Значение ЛОЖЬ представляется нулем. Результатом операции сравнения или логической операции являются ИСТИНА (true, 1) или ЛОЖЬ (false, 0).

Перечень логических операций следующий:

&&-логическое и (в данном случае рассматривается результат при вычислении одного и другого выражения одновременно);

||- логическое или (к рассмотрению берется результат, который был достигнут, хотя бы одного из двух выражений);

!- отрицание в выполняемых действиях;

Пример программы, в которой показаны приемы и варианты использования логических операций.

```
#include "iostream"
#include "cstdlib"
using namespace std;
int main()
{
   int a,b=3;
   cin>>a;
   return (a || b) && !(a && b);
}
```

Перечень операций сравнения следующий:

”<” – меньше чем указанное выражение или число;

”>” – больше чем указанное выражение или число;

”>=”-больше или равно;

”<=” – меньше или равно;

”==” – знак равенства в выражениях;

”!=” – не равно.

Пример программы, в которой показаны приемы и варианты использования операций сравнения.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
  int a,b,i;
  cin>>a>>b>>i;
  //Результатом работы операций сравнения
  //будет вывод на экран логической единицы
  //в случае успешного выполнения условия
  cout<<(a<=b)<<endl;
  cout<<(i!=0)<<endl;
  system("PAUSE");
  return 0;
}
```

Порядок выполнения операций сравнения и логических в выражениях следующий (приоритет при анализе выражений):

!
>
>=
<
<=,
==
!=
&&
||

Основное правило при использовании вышеперечисленных операций заключается в следующем: результатом любой операции сравнения или логической операции есть 0 или 1.

Следующий пример демонстрирует работу логических операций и операций сравнения.

```
#include <iostream>
```

```
#include <cstdlib>
#include <conio.h>
using namespace std;
int main()
{
    int n=5,j=0;
    cout<<!(n==5)<<endl;            //Выводится 0
    cout<<((n>=5)&&(n<100))<<endl; //Выводится 1
    cout<<((n>5)||(n<7))<<endl;    //Выводится 1
    cout<<!j<<endl;                //Выводится 1
    _getch();
    return 0;
}
```

Функция _getch() работает аналогичным образом как и getch(), которая была рассмотрена ранее. Употребление символа ”_” позволяет воспользоваться режимом совместимости по вызову указанной функции.

4.3. Поразрядные операции

Язык C++ в своей основе содержит операции, позволяющие оперировать с отдельными битами целочисленных типов. Это обусловлено тем, что лингвистика языка была задумана с таким подходом, чтобы во многих приложениях заменить ассемблер, который способен оперировать битами данных. Поразрядные операции — это тестирование (проверка), сдвиг или присвоение значений отдельным битам данных. Эти операции осуществляются над ячейками памяти, содержащими данные типа char или int. Данные типа float, double, long double, void или другие более сложные не могут участвовать в поразрядных операциях. Следующий список знаков поразрядных операций, выполняемых над отдельными разрядами (битами) операндов, представлен далее с рекомендациями по использованию.

&-битовое И. Проверяются поразрядно биты в двух целых числах и при включенных одновременно 2 битах значение конечного бита меняется на единицу (сохраняется).

```
&10101001
  11001010
  ----------
  10001000
```

В приведенном примере производится операция побитового сравнения чисел 169 и 202 в двоичном представлении - результатом является новое число 136. Следующая программа позволяет проверить полученный результат

```
#include <iostream>
#include <cstdlib>
int main()
{
  unsigned int a=169,b=202;
  std::cout<<(a&b)<<std::endl;
  system("PAUSE");
  return 0;
}
```

| -битовое ИЛИ. Проверяются поразрядно биты в двух целых числах и при включенном хотя бы одном из 2 битов значение конечного бита меняется на единицу (сохраняется).

```
| 10101001
  11001010
  ----------
  11101011
```

Результатом при значениях из предыдущего примера будет число 235.

^ -исключающее ИЛИ

Исключаются только одинаковые биты в двух числах.

```
^ 10101001
   11001010
  -----------
   01100011
#include <iostream>
```

```
#include <cstdlib>
int main()
{
  unsigned int a=169,b=202;
  std::cout<<(a^b)<<std::endl;
  system("PAUSE");
  return 0;
}
```

Результатом при значениях из предыдущего примера будет число 99.

~-НЕ (Инверсия исходного значения)

Для числа, т.е. число 10 будет преобразовано в -11.

```
  ~  00001010
     ----------
     11110101
```

В программе реализовано подобное преобразование.

```
#include <iostream>
#include <cstdlib>
int main()
{
   int a=10;
  std::cout<<(~a)<<std::endl;
  system("PAUSE");
  return 0;
}
```

>> -сдвиг вправо;

<<- сдвиг влево.

Операции позволяют посредством смещения битов вправо или влево заменять деление или умножение чисел на 2. Такие действия выполняются быстрее, поэтому рекомендуется применять указанные операции там, где это возможно.

Пример: сдвиг числа 15 на две позиции влево приведет к получению нового числа, равного 60.

```
#include <iostream>
#include <cstdlib>
int main()
{
  std::cout<<(15<<2)<<std::endl;
```

```
  system("PAUSE");
  return 0;
}
```

В следующем примере показано, как поменять местами значения переменных, не прибегая к промежуточной переменной.

```
#include <iostream>
#include <cstdlib>
int main()
{
int x=3, y=4;
x ^= y ^= x ^= y;
std::cout<<x<<std::endl;
std::cout<<y<<std::endl;
system("PAUSE");
};
```

Задачи на самостоятельное решение

1. Дано целое неотрицательное число. Поменять значение пятого по счету бита на обратное и вывести значение бита на экран ЭВМ.
2. Дано число 10. Не используя арифметических операций получить число 80.
3. Вывести максимальное число на экран ЭВМ из диапазона неотрицательных чисел типа int.

5. РАБОТА С УСЛОВНЫМИ КОНСТРУКЦИЯМИ И ОПЕРАТОРАМИ

Условные конструкции обрабатывают в программе такие ситуации, когда необходимо принять решение в зависимости от полученных расчетных или имеющихся данных. Результатом таких действий, как правило, является

получение некоторого логического значения, относительно которого производится ветвление в программе. Логических вариантов предоставляется два - true (единица или число, отличное от нуля) и false (ноль). При работе с условными конструкциями принято следующее соглашение: сразу же после проверки условия следует ветвление, содержащее решение в случае true. В противном случае выражение после условия игнорируется. Но в случае использования составной условной конструкции добавляется конструкция, в которой присутствует и альтернативное решение задачи. Более того, в условных конструкциях применятся и комбинированные способы обработки информации. Допустимо в программе использовать и несколько условных выражений, следующих друг за другом. В языке C++ существуют два условных оператора: if и switch. При определенных обстоятельствах оператор ? является альтернативой оператора if.

5.1. Условный оператор if

Выражение, использующее условную конструкцию if, выглядит следующим образом:

if (проверка условия) запись выражения;

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a,b;
    cin>>a>>b;
    if (a>b) cout<<a<<endl;
    system("PAUSE");
    return 0;
}
```

Однако в программе может быть реализовано условие с дополнительным ветвлением, которое выглядит следующим образом:

if (проверка условия) запись выражения1;
else запись выражения2;

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
    int a,b;
    cin>>a>>b;
    if (a>b) cout<<a<<endl;
    else cout<<b<<endl;
    system("PAUSE");
    return 0;
}
```

Допускается использование вложенных условий в основное. Оператор if является вложенным, если он вложен, т.е. находится внутри другого оператора if или else. В практике программирования вложенные условные операторы используются довольно часто. Во вложенном условном операторе фраза else всегда ассоциирована с ближайшим if в том же блоке, если этот if не ассоциирован с другой фразой else. Например:

```
if(i)
{
  if(j) параметр 1;
  if(k) параметр 2; // этот if
  else  параметр 3; // ассоциирован с этим else
}
else параметр 4; // ассоциирован с if(i)
```

Последняя фраза else не связана с if(j) потому, что она находится в другом блоке. Эта фраза else ассоциирована с if(i). Внутренняя фраза else ассоциирована с if(k), потому что этот if — ближайший.

Стандартом С89 допускает 15 уровней вложенности условных операторов, С99 — 127 уровней. В настоящее время

большинство компиляторов допускают значительно большее количество уровней вложенности. Однако на практике необходимость в глубине вложенности, большей, чем несколько уровней, возникает довольно редко, так как увеличение глубины вложенности быстро запутывает программу и делает ее нечитаемой.

Допускается использование нескольких выражений в одном ветвлении. Достигается это введением фигурных скобок и перечислением ряда выражений.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
  int i;
  cout<<"Enter a value:"<<endl;
  cin>>i;
  if(i>0)
  {
  i+=i;
  i++;
  cout<<"Add in C++: "<<i<<endl;
  }
  else
  {
  i-=i;
  i--;
  cout<<"Sub in C++: "<<i<<endl;
  }
  system("PAUSE");
  return 0;
}
```

5.2. Условный оператор "?"

Оператор ? можно использовать вместо оператора if-else, записанного в форме

```
if (условие) переменная = выражение;
else переменная = выражение;
```

Оператор ? является тернарным, потому что он имеет три операнда. Его общая форма записи в программе следующая:

```
Выражение1 ? Выражение2 : Выражение3;
```

Двоеточие разделяет два выражения в зависимости от от условия: true или false. Результат операции ? определяется следующим образом. Сначала вычисляется Выражение1. Если оно имеет значение ИСТИНА, вычисляется Выражение2 и его значение становится результатом операции ?. Если Выражение1 имеет значение ЛОЖЬ, вычисляется Выражение3 и его значение становится результатом операции ?. Например:

```
x = 10;
y = x>9 ? 100 : 200;
```

В этом примере переменной y присваивается значение 100. Если бы x было меньше 9, то переменная y получила бы значение 200. То же самое можно записать, используя оператор if-else:

```
x = 10;
if(x>9) y = 100;
else y = 200;
```

В следующем примере оператор ? используется для присвоения квадрату числа знака числа. (Само число вводится пользователем.) В этой программе при возведении в квадрат фактически сохраняется знак числа. Например, если пользователь введет 10, это число будет возведено в квадрат и в результате программа напечатает 100, а если пользователь введет число -10, то оно будет возведено в квадрат и

результату будет приписан знак числа; в этом случае будет напечатано -100.

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
  int isqrd, i;
  cout<<"Enter a value:"<<endl;
  cin>>i;
  isqrd = i>0 ? i*i : -(i*i);
  cout<<"Value in SQR: "<<i<<"=>"<<isqrd<<endl;
  system("PAUSE");
  return 0;
}
```

5.3. Оператор выбора – switch

Оператор выбора switch (часто его называют переключателем) предназначен для выбора ветви вычислительного процесса исходя из значения управляющего выражения. При этом значение управляющего выражения сравнивается со значениями в списке целых или символьных констант. Если будет найдено совпадение, то выполнится ассоциированный с совпавшей константой оператор. Общая форма оператора switch следующая:

```
switch (выражение) {
  case постоянная1:
    последовательность операторов
    break;
  case постоянная2:
    последовательность операторов
    break;
  case постоянная3:
    последовательность операторов
    break;
  default:
```

```
    последовательность операторов;
}
```

Значение выражения оператора switch должно быть таким, чтобы его можно было выразить целым числом. Это означает, что в управляющем выражении можно использовать переменные целого или символьного типа, но только не с плавающей точкой. Значение управляющего выражения по очереди сравнивается с постоянными величинами в операторах case. Если значение управляющего выражения совпадет с одной из постоянных, управление передается на соответствующую метку case и выполняется последовательность операторов до оператора break. Если оператор break отсутствует, выполнение последовательности операторов продолжается до тех пор, пока не встретится break (в другой метке) или не кончится тело оператора switch (т.е. блок, следующий за switch). Оператор default выполняется в том случае, когда значение управляющего выражения не совпало ни с одной постоянной. Оператор default также может отсутствовать. В этом случае при отсутствии совпадений не выполняется ни один оператор.

Оператор switch имеет следующие особенности:

- switch отличается от if тем, что в нем управляющее выражение проверяется только на равенство с постоянными, в то время как в if проверя ется любой вид отношения или логического выражения;

- в одном и том же операторе switch никакие два оператора case не могут иметь равных постоянных. Конечно, если один switch вложен в другой, в их операторах case могут быть совпадающие постоянные;

- если в управляющем выражении оператора switch встречаются символьные константы, они автоматически преобразуются к целому типу по принятым в языке C++ правилам приведения типов.

В следующем примере приведена реализация простейшего калькулятора с использованием switch().

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
  float x,y,z;
  char ch;
  cin>>y>>z;
  cout<<"Enter a math operation:"<<endl;
   cin>>ch;
  switch (ch)
  {
  case '-':
          x=y-z;
          break;
  case '+':
          x=y+z;
          break;
  case '*':
          x=y*z;
          break;
  case '/':
          x=y/z;
          break;
  default:
          cout<<"Symbol is errors";
          break;
      }
  cout<<"Result: "<<x<<endl;
  system("PAUSE");
  return 0;
  }
```

Схема выполнения оператора switch следующая:

- вычисляется выражение в круглых скобках;

- вычисленные значения последовательно сравниваются с константными выражениями, следующими за ключевыми словами case;

- если одно из константных выражений совпадает со значением выражения, то управление передается на оператор, помеченный соответствующим ключевым словом case;

- если ни одно из константных выражений не равно выражению, то управление передается на оператор, помеченный ключевым словом default, а в случае его отсутствия управление передается на следующий после switch оператор.

Задачи на самостоятельное решение

1. Даны три числа. Если их сумма равна одному числу из диапазона от 1 до 10, то 1 число вычесть из второго, если нет - умножить эти же числа.

2. Из введенных 5 значений определить наибольшее значение, не пользуясь конструкциями повторений (for и while)

3. Определить все числа, кратные 2, в диапазоне значений от 1 до 10. Конструкции с циклами не использовать.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Довбуш Г.Ф. Visual C++ на примерах / Г.Ф. Довбуш, А.Д. Хомоненко. – СПб.: БХВ-Петербург, 2007. – 528 с.
2. Пахомов Б.И. C/C++ и MS Visual C++ 2010 для начинающих / Б.И. Пахомов. – СПб.: БХВ-Петербург, 2011. – 736 с.
3. Мюссер Д.Р. C++ и STL: справочное руководство / Д.Р. Мюссер, Ж.Дж. Дердж, А. Сейни. 2-е изд. - М.: ООО ”И.Д. Вильямс”, 2010. – 430 с.
4. Прата С. Язык программирования C++. Лекции и упражнения / С. Прата. 5-е изд. – М.: ООО ”И.Д. Вильямс”, 2007. – 1184 с.
5. Страуструп Б Язык программирования C++ /Б. Страуструп. - М.: Бином, 2011. – 1136 с.
6. Шилдт Г. C++ Базовый курс / Г. Шилдт. 3-е изд. – М.: ООО ”И.Д. Вильямс”, 2010. – 624 с.
7. Марченко А.Л. C++. Бархатный путь / А.Л. Марченко. – М.: Горячая Линия - Телеком, 2005. – 400 с.
8. Коплиен Дж. Программирование на C++ / Дж. Коплиен. – СПб.: Питер, 2005. – 480 с.
9. Roberge J. A laboratory course in C++ structures. 2ed./ J. Roberge, S. Brandl, D. Whittington. Jones and Bartlett, 2003. -411 p.
10. London J. Modeling Derivatives in C++ / London J. Wiley, 2005. -841p.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ ……...1
1. РАЗРАБОТКА ПРОГРАММНЫХ СРЕДСТВ НА ЯЗЫКЕ С++…….……2
1.1. Общие сведения о языке разработки и интегрированных средствах создания программных проектов……...2
1.2. Разработка консольных приложений на С++……....3
2. ТИПЫ ДАННЫХ ……..…….……..……..…… 11
3. ПЕРЕМЕННЫЕ И КОНСТАНТЫ……...…….. 15
3.1. Локальные и глобальные переменные……..……..… 15
3.2. Константы и способы их задания……..……..... 18
4. ОПЕРАЦИИ И СИНТАКСИС ……..……..…….20
4.1. Арифметические операции …….……...…..21
4.2. Операции сравнения и логические операции …..…..……23
4.3. Поразрядные операции ……...……26
5. РАБОТА С УСЛОВНЫМИ КОНСТРУКЦИЯМИ И ОПЕРАТОРАМИ ……...……..29
5.1. Условный оператор if ……...…30
5.2. Условный оператор "?" …….32
5.3. Оператор выбора – switch …….34
БИБЛИОГРАФИЧЕСКИЙ СПИСОК …….…….. 38

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к практическим работам по дисциплине для студентов направления 230100.64 «Информатика и вычислительная техника» профиля «Системы автоматизированного проектирования в машиностроении» очной формы обучения
Часть 1

Составители
Юров Алексей Николаевич
Паринов Максим Викторович
Чижов Михаил Иванович
Рыжков Владимир Анатольевич

В авторской редакции

Компьютерный набор А.Н. Юрова

Подписано в изданию 10.11.2011.
Уч.-изд. л. 2,4. «С»

ФГБОУВПО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14