В серии: Библиотека ALT Linux

Программирование на языке C++ в среде Qt Creator

Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк,
О. В. Чеснокова, А. С. Чмыхало

Москва
ALT Linux
2014

УДК 004.43
ББК 32.973.26-018.1
А47

А47 Программирование на языке C++ в среде Qt Creator: / Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк,О. В. Чеснокова, А. С. Чмыхало — М. : ALT Linux, 2014. — 51 с. : ил. — (Библиотека ALT Linux).

ISBN 978-5-905167-16-4

Книга является учебником по алгоритмизации и программированию на C++ и пособием по разработке визуальных приложений в среде QT Creator. Также в книге описаны среда программирования Qt Creator, редактор Geany, кроссплатформенная билиотека построения графиков MathGL. При чтении книги не требуется предварительного знакомства с программированием.

Издание предназначено для студентов, аспирантов и преподавателей вузов, а также для всех, кто изучает программирование на C++ и осваивает кроссплатформенный инструментарий Qt для разработки программного обеспечения.

Сайт книги: http://www.altlinux.org/Books:Qt-C++

Книга адресована студентам, преподавателям инженерных и математических специальностей, а также научным работникам.

УДК 004.43
ББК 32.973.26-018.1

По вопросам приобретения обращаться: ООО «Альт Линукс»
(495)662-38-83 E-mail: sales@altlinux.ru http://altlinux.ru

ISBN 978-5-905167-16-4

Оглавление

Предисловие

Книга, которую открыл читатель, является с одной стороны учебником по алгоритмизации и программированию на C++, а с другой — пособием по разработке визуальных приложений в среде QT Creator. В книге описаны среда программирования Qt Creator и редактор Geany. При чтении книги не требуется предварительного знакомства с программированием.

В первой части книги (главы 1–9) на большом количестве примеров представлены методы построения программ на языке C++, особое внимание уделено построению циклических программ, программированию с использованием функций, массивов, матриц и указателей.

Вторая часть книги (глава 10) посвящена объектно-ориентированному программированию на C++.

В третьей части книги (главы 11–15) читатель научиться создавать кроссплатформенные визуальные приложения с помощью Qt Creator и познакомится с библиотекой классов Qt.

В книге присутствуют задания для самостоятельного решения.

Приложениях описан текстовый редактор Geany, а также кроссплатформенная библиотека MathGL предназначена для построения различных двух- и трёхмерных графиков.

Главы 1–9 написаны Е. Р. Алексеевым и О. В. Чесноковой. Автором раздела по объектно-ориентированному программированию является Д. А. Костюк. Главы 11–15, посвящённые программированию с использованием инструментария Qt, написаны Г. Г. Злобиным и А. С. Чмыхало.

Авторы благодарят компанию ALT Linux (www.altlinux.ru) и лично Алексея Смирнова и Владимира Чёрного за возможность издать очередную книгу по свободному программному обеспечению.

Глава 1

Знакомство с языком C++

В этой главе читатель напишет свои первые программы на языке C(C++), познакомится с основными этапами перевода программы с языка C++ в машинный код. Второй параграф главы посвящён знакомству со средой Qt Creator.

1.1 Первая программа на C++

Знакомство с языком C++ начнём с написания программ, предназначенных для решения нескольких несложных задач.

Задача 1.1. Заданы две стороны прямоугольника a, b. Найти его площадь и периметр.

Как известно, периметр прямоугольника $P = 2 \cdot (a + b)$, а его площадь $S = a \cdot b$. Ниже приведён текст программы.

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      float a,b,s,p;
6      cout<<"a=";
7      cin>>a;
8      cout<<"b=";
9      cin>>b;
10     p=2*(a+b);
11     s=a*b;
12     cout << "Периметр прямоугольника равен " << p <<endl;
13     cout << "Площадь прямоугольника равна " << s <<endl;
14     return 0;
15 }
```

Давайте построчно рассмотрим текст программы и познакомимся со структурой программы на C++ и с некоторыми операторами языка.

Строка 1. Указывает компилятору (а точнее, препроцессору), что надо использовать функции из стандартной библиотеки `iostream`. Библиотека `iostream` нужна для организации ввода с помощью инструкции `cin` и вывода — с помощью `cout`. В программе на языке C++ должны быть подключены все используемые библиотеки.

Строка 2. Эта строка обозначает, что при вводе и выводе с помощью `cin` и `cout` будут использоваться стандартные устройства (клавиатура и экран), если эту строку не указывать, то каждый раз при вводе вместо `cin` надо будет писать `std::cin`, а вместо `cout` — `std::cout`.

Строка 3. Заголовок главной функции (главная функция имеет имя `main`). В простых программах присутствует только функция `main()`.

Строка 4. Любая функция начинается с символа `{`.

Строка 5. Описание вещественных (`float`) переменных `a` (длина одной стороны прямоугольника), `b` (длина второй стороны прямоугольника), `s` (площадь прямоугольника), `p` (периметр прямоугольника). Имя переменной[1] состоит из латинских букв, цифр и символа подчёркивания. Имя не может начинаться с цифры. В языке C++ большие и малые буквы различимы. Например, имена `PR_1`, `pr_1`, `Pr_1` и `pR_1` — разные.

Строка 6. Вывод строки символов `a=` с помощью `cout`. Программа выведет подсказку пользователю, что необходимо вводить переменную `a`

Строка 7. Ввод вещественного числа `a` с помощью `cin`. В это момент программа останавливается и ждёт, пока пользователь введёт значение переменой `a` с клавиатуры.

Строка 8. Вывод строки символов `b=` с помощью `cout`.

Строка 9. Ввод вещественного числа `b` с помощью `cin`.

Строка 10. Оператор присваивания для вычисления периметра прямоугольника (переменная `p`) по формуле $2 \cdot (a + b)$. В операторе присваивания могут использоваться круглые скобки и знаки операций: + (сложение), - (вычитание), * (умножение), / (деление).

Строка 11. Оператор присваивания для вычисления площади прямоугольника.

[1] В литературе равнозначно используются термины «имя переменной» и «идентификатор».

Строка 12. Вывод строки «Периметр прямоугольника равен » и значения p на экран. Константа endl хранит строку «\n», которая предназначена для перевода курсора в новую строку дисплея[2]. Таким образом строка

```
cout << "Периметр прямоугольника равен "<< p <<endl;
```

выводит на экран текст "Периметр прямоугольника равен "[3], значение переменной p, и переводит курсор в новую строку.

Строка 13. Вывод строки "Площадь прямоугольника равна ", значения площади прямоугольника s, после чего курсор переводится в новую строку дисплея.

Строка 14. Оператор return, который возвращает значение в операционную систему. Об этом подробный разговор предстоит в п. ?? Сейчас следует запомнить, если программа начинается со строки int main(), последним оператором должен быть return 0[4].

Строка 15. Любая функция (в том числе и main) заканчивается символом }.

Мы рассмотрели простейшую программу на языке C++, состоящую из операторов ввода данных, операторов присваивания (в которых происходит расчет по формулам) и операторов вывода.

Любая программа на языке C++ представляет собой одну или несколько функций. В любой программе **обязательно** должна быть одна функция main(). С этой функции начинается выполнение программы. Правилом хорошего тона в программировании является разбиение задачи на подзадачи, и в главной функции чаще всего должны быть операторы вызова других функций. Общую структуру любой программы на языке C++ можно записать следующим образом.

```
Директивы препроцессора
Объявление глобальных переменных
Тип_результата  f1 (Список_переменных)
{
Операторы
}
Тип_результата  f2 (Список_переменных)
{
Операторы
}
```

[2]Обращаем внимание читателя, что символ пробел является обычным символом, который ничем не отличается от остальных. Для вывода пробела на экран его надо явно указывать в строке вывода.

[3]С пробелом после слова «равен».

[4]Вообще говоря, вместо нуля может быть любое целое число.

```
...

Тип_результата fn(Список_переменных)
{
Операторы
}
Тип_ результата main(Список_переменных)
{
Операторы
}
```

На первом этапе знакомства с языком мы будем писать программы, состоящие только из функции main, без использования глобальных переменных. Структура самой простой на C(C++) имеет вид.

```
Директивы препроцессора
Тип_ результата main(Список_переменных)
{
Операторы
}
```

Введенная в компьютер программа на языке C++ должна быть переведена в двоичный машинный код (должен быть сформирован исполняемый файл). Для этого существуют специальные программы, называемые трансляторами. Все трансляторы делятся на два класса:

- *интерпретаторы* — трансляторы, которые переводят каждый оператор программы в машинный код, и по мере перевода операторы выполняются процессором;
- *компиляторы* переводят всю программу целиком, и если перевод всей программы прошел без ошибок, то полученный двоичный код можно запускать на выполнение.

Процесс перевода программы в машинный код называется *трансляцией*. Если в качестве транслятора выступает компилятор, то используют термин *компиляция* программы. При переводе программы с языка C++ в машинный код используются именно компиляторы, и поэтому применительно к языку C++ термины «компилятор» и «транслятор» эквивалентны.

Рассмотрим основные этапы обработки компилятором программы на языке C++ и формирования машинного кода.

1. Сначала программа обрабатывается препроцессором[5], который обрабатывает директивы препроцессора, в нашем случае это ди-

[5]Препроцессор — это программа, которая преобразовывает текст директив препроцессора в форму, понятную компилятору. О данных на выходе препроцессора говорят, что они находятся в препроцессированной форме.

ративы включения заголовочных файлов (файлов с расширением **.h**) — текстовых файлов, в которых содержится описание используемых библиотек. В результате формируется полный текст программы, который поступает на вход компилятора.

2. Компилятор разбирает текст программ на составляющие элементы, проверяет синтаксические ошибки и в случае их отсутствия формирует объектный код (файл с расширением **.o** или **.obj**). Получаемый на этом этапе двоичный код не включает в себя двоичные коды библиотечных функций и функций пользователя.
3. *Компоновщик* подключает к объектному коду программы объектные модули библиотек и других файлов (если программа состоит из нескольких файлов) и генерирует исполняемый код программы (двоичный файл), который уже можно запускать на выполнение. Этот этап называется компоновкой или сборкой программы.

После написания программы ее необходимо ввести в компьютер. В той книге будет рассматриваться работа на языке C++ в среде Qt Creator[6]. Поэтому перед вводом программы в компьютер надо познакомиться со средой программирования.

1.2 Среда программирования Qt Creator

Среда программирования Qt Creator (IDE IDE QT Creator) находится в репозитории большинства современных дистрибутивов Linux (ОС Linux Debian, ОС Linux Ubuntu, ОС ROSA Linux, ALT Linux и др.). Установка осуществляется штатными средствами вашей операционной системы (менеджер пакетов Synaptic и др.) из репозитория, достаточно установить пакет qtcreator, необходимые пакеты и библиотеки будут доставлены автоматически. Последнюю версию IDE Qt Creator можно скачать на сайте QtProject (http://qt-project.org/downloads). Скачанный установочный файл имеет расширение **.run**. Для установки приложения, необходимо запустить его на выполнение. Установка проходит в графическом режиме. После запуска програм-

[6]Тексты программ, приведённые в первой части книги (главы 1–9), без серьёзных изменений могут быть откомпилированы с помощью любого современного компилятора с языка C(C++). Авторы протестировали все программы из первой части книги с помощью QT Creator и IDE Geany (с использованием g++ версии 4.8).

мы пользователь увидит на экране окно, подобное представленному на рис. 1.1[7].

Рис. 1.1. Окно `Qt Creator`

При работе в `Qt Creator` вы находитесь в одном из режимов:

1. **Welcome** (Начало) — отображает экран приветствия, позволяя быстро загружать недавние сессии или отдельные проекты. Этот режим можно увидеть при запуске `Qt Creator` без указания ключей командной строки.
2. **Edit** (Редактор) — позволяет редактировать файлы проекта и исходных кодов. Боковая панель слева предоставляет различные виды для перемещения между файлами.
3. **Debug** (Отладка) — предоставляет различные способы для просмотра состояния программы при отладке;
4. **Projects** (Проекты) — используется для настройки сборки, запуска и редактирования кода.
5. **Analyze** (Анализ) — в Qt интегрированы современные средства анализа кода разрабатываемого приложения.

[7]Окно на вашем компьютере визуально может несколько отличаться от представленного на рис. 1.1, авторы использовали IDE `Qt Creator` версии 2.6.2, основанную на QT 5.0.1.

6. **Help** (Справка) — используется для вывода документации библиотеки Qt и Qt Creator.
7. **Output** (Вывод) — используется для вывода подробных сведений о проекте.

Рассмотрим простейшие приёмы работы в среде Qt Creator на примере создания консольного приложения для решения задачи 1.1. Для этого можно поступить одним из способов:

1. В меню **File** (Файл) выбрать команду **New File or Project** (Новый файл или проект) (комбинация клавиш **Ctrl**+**N**).
2. Находясь в режиме **Welcome** (Начало) главного окна QtCreator (рис. 1.1) щёлкаем по ссылке **Develop** (Разработка) и выбираем команду Create Project (Создать проект).

После это откроется окно, подобное представленному на рис. 1.2. Для создания простейшего консольного приложения выбираем **Non-Qt Project** (Проект без использования Qt) — **Plain C++ Project** (Простой проект на языке C++).

Далее выбираем имя проекта и каталог для его размещения (см. рис. 1.3)[8]. Следующие два этапа создания нашего первого приложения оставляем без изменения[9]. После чего окно IDE Qt Creator примет вид, подобное, представленное на рис. 1.4. Заменим текст текста стандартного приложения, которое выводит текст «Hello, Word», на текст программы решения задачи 1.1.

Для сохранения текста программы можно воспользоваться **Сохранить** или **Сохранить всё** из меню **Файл**. Откомпилировать и запустить программу можно одним из следующих способов:

1. Пункт меню **Сборка-Запустить**.
2. Нажать на клавиатуре комбинацию клавиш Ctrk+R.
3. Щёлкнуть по кнопке Запустить (▶).

Окно с результатами работы программы представлено на рис. 1.5.

Авторы сталкивались с тем, что в некоторых дистрибутивах Ubuntu Linux и Linux Mint после установки Qt Creator не запускались консольные приложения. Если читатель столкнулся с подобной

[8] Рекомендуем для каждого (даже самого простого) проекта выбирать отдельный каталог. Даже самый простой проект — это несколько взаимосвязанных между собой файлов и каталогов.

[9] О назначении этих этапов будет рассказано в дальнейших разделах книги.

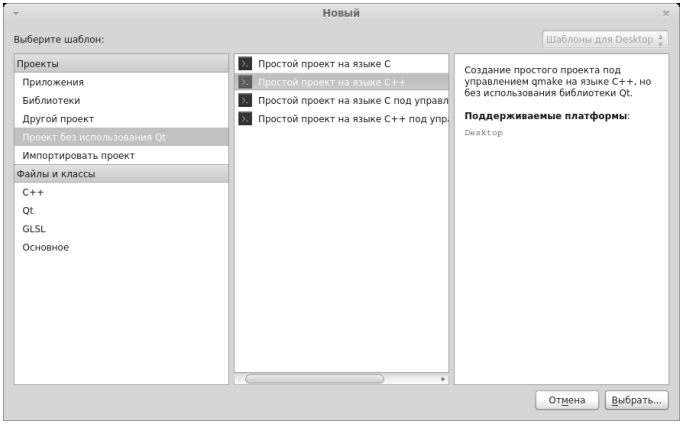

Рис. 1.2. Окно выбора типа приложения в `Qt Creator`

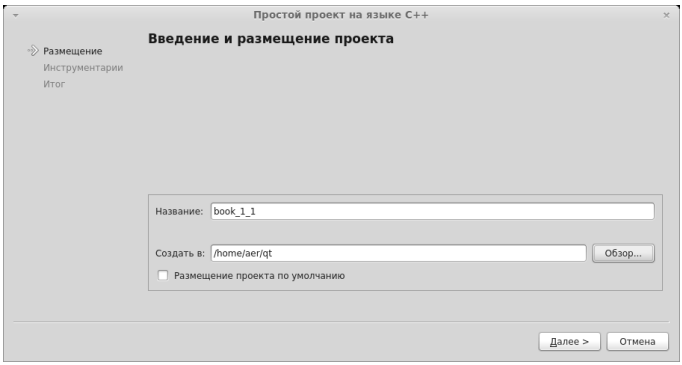

Рис. 1.3. Выбор имени и каталога нового проекта

проблемой, скорее всего надо корректно настроить терминал, который отвечает за запуск приложений в консоли. Для этого вызываем команду Tools — Options — Environment (см. рис. 1.6). Параметр **Terminal** (Терминал) должен быть таким же, как показано на рис. 1.6. Проверьте установлен ли в Вашей системе пакет xterm, и при

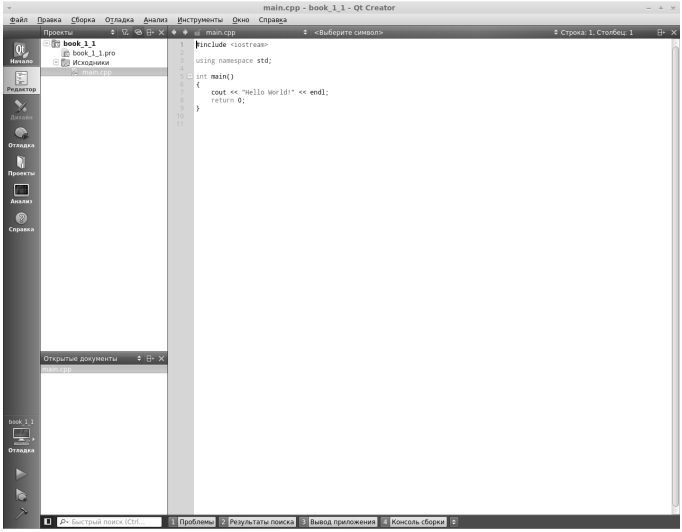

Рис. 1.4. Главное окно создания консольного приложения

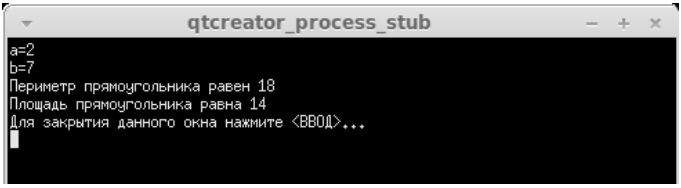

Рис. 1.5. Результаты работы программы решения задачи 1.1

необходимости доставьте его. После этого не должно быть проблем с запуском консольных приложений.

Аналогичным образом можно создавать и запускать любое консольное приложение.

Дальнейшее знакомство со средой Qt Creator продолжим, решая следующую задачу.

Задача 1.2. Заданы длины трёх сторон треугольника a, b и c (см. рис. 1.7). Вычислить площадь и периметр треугольника

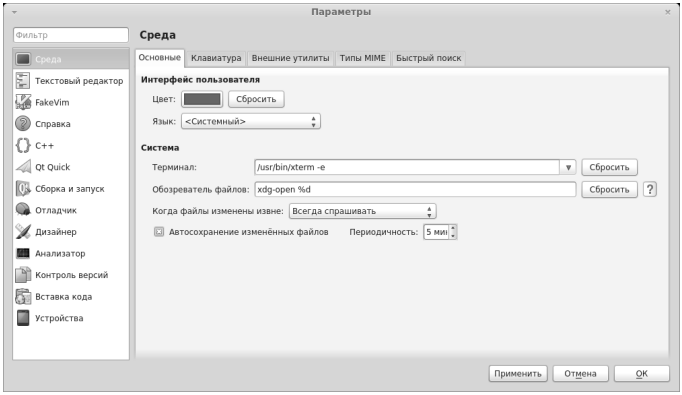

Рис. 1.6. Окно настроек среды `Qt Creator`

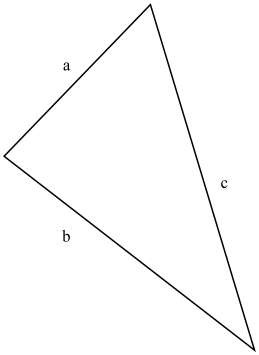

Рис. 1.7. Треугольник

Для решения задачи нам понадобится формула вычисления периметра $p = a + b + c$. Для вычисления площади можно воспользоваться формулой Герона $S = \sqrt{\frac{p}{2}\left(\frac{p}{2} - a\right)\left(\frac{p}{2} - b\right)\left(\frac{p}{2} - a\right)}$.

Решение задачи можно разбить на следующие этапы:

1. Определение значений a, b и c (ввод величин a, b, c с клавиатуры в память компьютера).
2. Расчет значений p и s по приведенным выше формулам.

3. Вывод p и s на экран дисплея.

Ниже приведен текст программы. Сразу заметим, что в тексте могут встречаться строки, начинающие с двух наклонных (//), являющиеся комментариями. *Комментарии* не являются обязательными элементами программы и ничего не сообщают компьютеру, они поясняют человеку, читающему текст программы, назначение отдельных элементов программы. В книге комментарии будут широко использоваться для пояснения отдельных участков программы.

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    float a,b,c,s,p;
    cout<<"Введите длины сторон треугольника"<<endl;
//Ввод значений длин треугольника a, b, c.
    cin>>a>>b>>c;
//Вычисление периметра треугольника.
    p=a+b+c;
//Вычисление площади треугольника.
    s=sqrt(p/2*(p/2-a)*(p/2-b)*(p/2-c));
//Вывод на экран дисплея значений площади и периметра треугольника.
    cout<<"Периметр треугольника равен "<<p<<", его площадь
        равна "<<s<<endl;
    return 0;
}
```

Кроме используемой в предыдущей программе библиотеки `iostream`, в строке 2 подключим библиотеку `math.h`, которая служит для использования математических функций языка C(C++). В данной программе используется функция извлечения квадратного корня — *sqrt(x)*. Остальные операторы (ввода, вывода, вычисления значений переменных аналогичны используемым в предыдущей программе.

Таким образом, выше были рассмотрены самые простые программы (линейной структуры), которые предназначены для ввода исходных данных расчёта по формулам и вывода результатов.

Глава 2

Общие сведения о языке C++

В этой главе читатель познакомится с основными элементами языка C++: алфавитом, переменными, константами, типами данных, основными операциями, стандартными функциями, структурой программы и средствами ввода вывода данных.

2.1 Алфавит языка

Программа на языке C++ может содержать следующие символы:

- прописные, строчные латинские буквы A, B, C, . . . , x, y, z и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки: “ { } , | [ ] ( ) + — / % * . \ ‘ : ? < = > ! & # ~ ; ^
- символы пробела, табуляции и перехода на новую строку.

Из символов алфавита формируют ключевые слова и идентификаторы. Ключевые слова — это зарезервированные слова, которые имеют специальное значение для компилятора и используются только в том смысле, в котором они определены (операторы языка, типы данных и т.п.). Идентификатор — это имя программного объекта, представляющее собой совокупность букв, цифр и символа подчеркивания. Первый символ идентификатора — буква или знак подчеркивания, но не цифра. Идентификатор не может содержать пробел. Прописные и строчные буквы в именах различаются, например, `ABC`, `abc`, `Abc` — три различных имени. Каждое имя (идентификатор) должно быть уникальным в пределах функции и не совпадать с ключевыми словами.

В тексте программы можно использовать комментарии. Если текст начинается с двух символов «косая черта» // и заканчивается символом перехода на новую строку или заключен между символами /* и */, то компилятор его игнорирует.

```
/* Комментарий можетвыглядеть так!*/

//А если вы используете такой способ,
//то каждая строка должна начинаться
//с двух символов «косая черта».
```

Комментарии удобно использовать как для пояснений к программе, так и для временного исключения фрагментов программы при отладке.

2.2 Данные

Для решения задачи в любой программе выполняется обработка каких-либо данных. Данные хранятся в памяти компьютера и могут быть самых различных типов: целыми и вещественными числами, символами, строками, массивами. Типы данных определяют способ хранения чисел или символов в памяти компьютера. Они задают размер ячейки, в которую будет записано то или иное значение, определяя тем самым его максимальную величину или точность задания. Участок памяти (ячейка), в котором хранится значение определенного типа, называется переменной. У переменной есть имя (идентификатор) и значение. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменить. Перед использованием любая переменная должна быть описана. Оператор описания переменных в языке C++ имеет вид:

```
тип имя_переменной;
```

или

```
тип список_переменных;
```

Типы данных языка C++ можно разделить на основные и составные.

К основным типам данных языка относят:

- `char` — символьный;
- `int` — целый;
- `float` — с плавающей точкой;
- `double` — двойной точности;
- `bool` — логический.

Для формирования других типов данных используют основные типы и так называемые спецификаторы. Типы данных, созданные на базе стандартных типов с использованием спецификаторов, называют составными типами данных. В C++ определены четыре спецификатора типов данных:

- `short` — короткий;
- `long` — длинный;
- `signed` — знаковый;
- `unsigned` — беззнаковый.

Далее будут рассмотрены назначение и описание каждого типа.

2.2.1 Символьный тип

Данные типа `char` в памяти компьютера всегда занимают один байт[1]. Это связано с тем, что обычно под величину символьного типа отводят столько памяти, сколько необходимо для хранения любого из 256 символов клавиатуры. Символьный тип может быть со знаком или без знака (табл. 2.1).

Таблица 2.1: Символьные типы данных

Тип	**Диапазон**	**Размер**
`char`	–128...127	1 байт
`unsigned char`	0...255	1 байт
`signed char`	–128...127	1 байт

Пример описания символьных переменных:

```
char c, str; //Описаны две символьные переменные.
```

При работе с символьными данными нужно помнить, что если в выражении встречается одиночный символ, он должен быть заключен в одинарные кавычки. Например, `'a'`, `'b'`, `'+'`, `'3'`.

Последовательность символов, то есть строка, при использовании в выражениях заключается в двойные кавычки: `"Hello!"`.

2.2.2 Целочисленный тип

Переменная типа `int` в памяти компьютера может занимать либо два, либо четыре байта. Это зависит от разрядности процессора.

[1]В кодировке utf каждый символ кириллицы занимает 2 байта.

Диапазоны значений целого типа представлены в таблице 2.2. По умолчанию все целые типы считаются знаковыми, т.е. спецификатор `signed` можно не указывать.

Таблица 2.2: Целые типы данных

Тип	**Диапазон**	**Размер**
`int`	–2147483647 ... 2147483647	4 байта
`unsigned int`	0 ... 4294967295	4 байта
`signed int`	–2147483647 ... 2147483647	4 байта
`short int`	–32767 ... 32767	2 байта
`long int`	–2147483647 ... 2147483647	4 байта
`unsigned short int`	0 ... 65535	2 байта
`signed short int`	–32767 ... 32767	2 байта
`long long int`	$-(2^{63}-1)$ ... $(2^{63}-1)$	8 байт
`signed long int`	–2147483647 ... 2147483647	4 байта
`unsigned long int`	0 ... 4294967295	4 байта
`unsigned long long int`	0 ... $2^{64}-1$	8 байт

Пример описания целочисленных данных:

```
int a, b, c;
unsigned long int A, B, C;
```

2.2.3 Вещественный тип

Внутреннее представление вещественного числа в памяти компьютера отличается от представления целого числа. Число с плавающей точкой представлено в экспоненциальной форме $mE \pm p$, где m — мантисса (целое или дробное число с десятичной точкой), p — порядок (целое число). Для того чтобы перевести число в экспоненциальной форме к обычному представлению с фиксированной точкой, необходимо мантиссу умножить на десять в степени порядок. Например,

$-6.42E + 2 = -6.42 \cdot 10^2 = -642$,

$3.2E - 6 = 3.2 \cdot 10^{-6} = 0.0000032$

Обычно величины типа `float` занимают 4 байта, из которых один двоичный разряд отводится под знак, 8 разрядов под порядок и 23 под мантиссу. Поскольку старшая цифра мантиссы всегда равна 1, она не хранится.

Величины типа `double` занимают 8 байт, в них под порядок и мантиссу отводится 11 и 52 разряда соответственно. Длина мантиссы определяет точность числа, а дли-на порядка его диапазон. Спецификатор типа `long` перед именем типа `double` указывает, что под величину отводится 10 байт.

Диапазоны значений вещественного типа представлены в таблице 2.3.

Таблица 2.3: Вещественные типы данных

Тип	Диапазон	Размер
`float`	3.4E-38 ... 3.4E+38	4 байта
`double`	1.7E-308 ... 1.7E+308	8 байт
`long double`	3.4E-4932 ... 3.4E+4932	10 байт

Пример описания вещественных переменных:

```
double x1,x2,x3;
float X, Y, Z;
```

2.2.4 Логический тип

Переменная типа `bool` может принимать только два значения `true` (истина) или `false` (ложь). Любое значение не равное нулю интерпретируется как `true`, а при преобразовании к целому типу принимает значение равное 1. Значение `false` представлено в памяти как 0.

Пример описания данных логического типа:

```
bool F, T;
```

2.2.5 Тип void

Множество значений этого типа пусто. Он используется для определения функций, которые не возвращают значения, для указания пустого списка аргументов функции, как базовый тип для указателей и в операции приведения типов.

2.3 Константы

Константы это величины, которые не изменяют своего значения в процессе выполнения программы. Оператор описания константы имеет вид:

```
const тип имя_константы = значение;
```

Константы в языке C++ могут быть целыми, вещественными, символьными или строковыми. Обычно компилятор определяет тип константы по внешнему виду, но существует возможность и явного указания типа, например,

```
const double pi=3.141592653589793
```

Кроме того, константа может быть определена с помощью директивы[2] `#define`. Эта директива служит для замены часто использующихся констант, ключевых слов, операторов или выражений некоторыми идентификаторами. Идентификаторы, заменяющие текстовые или числовые константы, называют именованными константами. Основная форма синтаксиса директивы следующая:

```
#define идентификатор текст
```

Например,

```
#define PI 3.141592653589793
int main()
...
```

2.4 Структурированные типы данных

Структурированный тип данных характеризуется множественностью образующих его элементов. В C++ это массивы, строки, структуры и файлы.

Массив — совокупность данных одного и того же типа[3]. Число элементов массива фиксируется при описании типа и в процессе выполнения программы не изменяется.

В общем виде массив можно описать так:

```
тип имя [размерность_1][размерность_2]...[размерность_N];
```

Например,

```
float x[10];     //Описан массив из 10 целых чисел.
int a[3][4];     //Описан двумерный массив,
              //матрица из 3-х строк и 4-х столбцов.
double b[2][3][2];  //Описан трехмерный массив.
```

Для доступа к элементу массива достаточно указать его порядковый номер, а если массив многомерный (например, таблица), то несколько номеров:

[2]Структура программы и директивы описаны в п. ??

[3]Подробно работа с одномерными и двумерными массивами описана в главах ?? и ??.

имя_массива[номер_1][номер_2]...[номер_N]

Например: x[5], a[2][3], b[1][2][2].

Элементы массива в C++ нумеруются с нуля. Первый элемент, всегда имеет номер ноль, а номер последнего элемента на единицу меньше заданной при его описании размерности:

```
char C[5];  //Описан массив из 5 символов,
           //нумерация от 0 до 4.
```

Строка — последовательность символов[4]. В C++ строки описывают как массив элементов типа char. Например:

```
char s[25];   // Описана строка из 25 символов.
```

Структура[5] это тип данных, который позволяет объединить разнородные данные и обрабатывать их как единое целое.

Например

```
struct fraction  //Объявлена структура правильная дробь.
{
        //Определяем поля структуры:
  int num;    // поле числитель,
  int den;    // поле знаменатель.
}
...
fraction d, D[20];   //Определена переменная d,
                    //массив D[20], типа fraction.
...
d.num;        //Обращение к полю num переменной d.
D[4].den;    //Обращение к полю den
            //четвертого элемента массива D.
```

2.5 Указатели

Указатели широко применяются в C++. Можно сказать, что именно наличие указателей сделало этот язык удобным для системного программирования. С другой стороны это одна из наиболее сложных для освоения возможностей C++. Идея работы с указателями состоит в том, что пользователь работает с адресом ячейки памяти.

Как правило, при обработке оператора объявления переменной

тип имя_переменной;

[4]Работа со строками описана в главе ??

[5]Работа со структурами описана в главе ??.

компилятор автоматически выделяет память под переменную имя_переменной в соответствии с указанным типом:

```
char C; //Выделена память под символьную переменную C
```

Доступ к объявленной переменной осуществляется по ее имени. При этом все обращения к переменной заменяются на адрес ячейки памяти, в которой хранится ее значение. При завершении программы или функции, в которой была описана переменная, память автоматически освобождается.

Доступ к значению переменной можно получить иным способом — определить собственные переменные для хранения адресов памяти. Такие переменные называют указателями. С помощью указателей можно обрабатывать массивы, строки и структуры, создавать новые переменные в процессе выполнения программы, передавать адреса фактических параметров функциям и адреса функций в качестве параметров.

Итак, указатель это переменная, значением которой является адрес памяти, по которому хранится объект определенного типа (другая переменная). Например, если C это переменная типа char, а P — указатель на C, значит в P находится адрес по которому в памяти компьютера хранится значение переменной C.

Как и любая переменная, указатель должен быть объявлен. При объявлении указателей всегда указывается тип объекта, который будет храниться по данному адресу:

```
тип *имя_переменной;
Например:
int *p;     //По адресу, записанному в переменной p,
        //будет храниться переменная типа int
```

Звездочка в описании указателя, относится непосредственно к имени, поэтому чтобы объявить несколько указателей, ее ставят перед именем каждого из них:

```
float *x, y, *z;   //Описаны указатели на вещественное
//число -- x и z,само вещественное значение -- *x, *z.
//а так же вещественная переменная y, её адрес -- &y.
```

2.6 Операции и выражения

Выражение задает порядок выполнения действий над данными и состоит из операндов (констант, переменных, обращений к функциям), круглых скобок и знаков операций. Операции делятся на унар-

ные (например, -c) и бинарные (например, a+b). В таблице 2.4 представлены основные операции языка C++.

Таблица 2.4: Основные операции языка C++

Операция	Описание
Унарные операции	
++	увеличение значения на единицу
--	уменьшение значения на единицу
~	поразрядное отрицание
!	логическое отрицание
-	арифметическое отрицание (унарный минус)
+	унарный плюс
&	взятие адреса
*	разадресация
(type)	преобразование типа
Бинарные операции	
+	сложение
-	вычитание
*	умножение
/	деление
%	остаток от деления
<<	сдвиг влево
>>	сдвиг вправо
<	меньше
>	больше
<=	меньше или равно
>=	больше или равно
==	равно
!=	не равно
&	поразрядная конъюнкция (И)
^	поразрядное исключающее ИЛИ
|	поразрядная дизъюнкция (ИЛИ)
&&	логическое И
||	логическое ИЛИ
=	присваивание
*=	умножение с присваиванием
/=	деление с присваиванием

Таблица 2.4 — продолжение

Операция	Описание
+=	сложение с присваиванием
-=	вычитание с присваиванием
%=	остаток от деления с присваиванием
<<=	сдвиг влево с присваиванием
>>=	сдвиг вправо с присваиванием
&=	поразрядная конъюнкция с присваиванием
|=	поразрядная дизъюнкция с присваиванием
^=	поразрядное исключающее ИЛИ с присваиванием
Другие операции	
?	условная операция
,	последовательное вычисление
sizeof	определение размера
(тип)	преобразование типа

Перейдем к подробному рассмотрению основных операций языка.

2.6.1 Операции присваивания

Обычная операция присваивания имеет вид:

`имя_переменной=значение;`

где `значение` это выражение, переменная, константа или функция. Выполняется операция так. Сначала вычисляется значение выражения указанного в правой части оператора, а затем его результат записывается в область памяти, имя которой указано слева.

Например,

```
b=3;    //Переменной b присваивается значение равное трем.
a=b;    // Переменной a присваивается значение b.
c=a+2*b; // Переменной c присваивается значение выражения.
c=c+1;  // Значение переменой c увеличивается на единицу.
a=a*3;  // Значение переменой a увеличивается в три раза.
```

Задача 2.1. Пусть в переменной `a` хранится значение равное 3, а в переменную `b` записали число 5. Поменять местами значения переменных `a` и `b`.

Для решения задачи понадобится дополнительная переменная `c` (см. рис. 2.1). В ней временно сохраняется значение переменной `a`. Затем, значение переменной `b` записывается в переменную `a`, а значение переменной `c` в переменную `b`.

```
c=a;    // Шаг 1. c=3
a=b;    // Шаг 2. a=5
b=c;    // Шаг 3. b=3
```

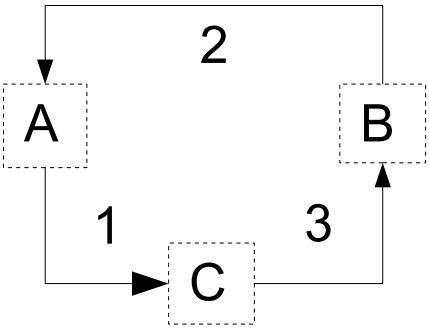

Рис. 2.1. Использование буферной переменной

Если в операторе присваивания левая и правая часть это переменные разных типов, то *происходит преобразование*: значение переменной в правой части преобразуется к типу переменной в левой части. Следует учитывать, что при этом можно потерять информацию или получить другое значение.

В C++ существует возможность присваивания нескольким переменным одного и того же значения. Такая операция называется *множественным присваиванием* и в общем виде может быть записана так:

```
имя_1 = имя_2 = ... = имя_N = значение;
```

Запись `a=b=c=3.14159/6;`

означает, что переменным `a`, `b` и `c` было присвоено одно и то же значение `3.14159/6`.

Операции $+=$, $-=$, $*=$, $/=$ называют *составным присваиванием*. В таких операциях при вычислении выражения стоящего справа используется значение переменной из левой части, например так:

```
x+=p; //Увеличение x на p, то же что и x=x+p.
x-=p; //Уменьшения x на p, то же что и x=x-p.
x*=p; //Умножение x на p, то же что и x=x*p.
x/=p; //Деление x на p, то же что и x=x/p.
```

2.6.2 Арифметические операции

Операции +, -, *, / относят к *арифметическим операциям*. Их назначение понятно и не требует дополнительных пояснений. При программировании арифметических выражений следует придерживаться простых правил. Соблюдать очерёдность выполнения арифметических операций. Сначала выполняются операции умножения и деления (1-й уровень), а затем сложения и вычитания (2-й уровень). Операции одного уровня выполняются последовательно друг за другом. Для изменения очерёдности выполнения операций используют скобки. Таблица 2.5 содержит примеры записи алгебраических выражений.

Таблица 2.5: Примеры записи алгебраических выражений

Математическая запись	**Запись на языке** C++
$2 \cdot a + b \cdot (c + d)$	`2*a+b*(c+d)`
$3 \cdot \dfrac{a+b}{c+d}$	`3*(a+b)/(c+d)`
$\dfrac{3 \cdot a - 2 \cdot b}{c \cdot d}$	`(3*a-2*b)/(c*d)` или `(3*a-2*b)/c/d`
$\dfrac{(b-a)^2}{c + \dfrac{1}{d-2}} - \dfrac{a^2+1}{b^2+cd}$	`(b-a)*(b-a)/(c+1/(d-2))-(a*a+1)/(b*b+c*d)`

Операции *инкремента* ++ и *декремента* -- так же причисляют к арифметическим, так как они выполняют увеличение и уменьшение на единицу значения переменной. Эти операции имеют две формы записи: префиксную (операция записывается перед операндом) и постфиксную (операция записывается после операнда). Так, например оператор `p=p+1;` можно представить в префиксной форме `++p;` и в постфиксной `p++;`. Эти формы отличаются при использовании их в выражении. Если знак декремента (инкремента) предшествует операнду, то сначала выполняется увеличение (уменьшение) значения операнда, а затем операнд участвует в выражении. Например,

```
x=12;
y=++x; //В переменных x и y будет храниться значение 13.
```

Если знак декремента (инкремента) следует после операнда, то сначала операнд участвует в выражении, а затем выполняется увеличение (уменьшение) значения операнда:

```
x=12;
y=x++; //Результат — число 12 в переменной y, а в x — 13.
```

Остановимся на *операциях целочисленной арифметики*.

Операция целочисленного деления / возвращает целую часть частного (дробная часть отбрасывается) в том случае если она применяется к целочисленным операндам, в противном случае выполняется обычное деление: 11/4=2 или 11.0/4=2.75.

Операция остаток от деления % применяется только к целочисленным операндам: 11%4 = 3.

К *операциям битовой арифметики* относятся следующие операции: &, |, ^, ~, <<, >>. В операциях битовой арифметики действия происходят над двоичным представлением целых чисел.

Арифметическое И (&). Оба операнда переводятся в двоичную систему, затем над ними происходит логическое поразрядное умножение операндов по следующим правилам:

1&1=1, 1&0=0, 0&1=0, 0&0=0.

Например, если A=14 и B=24, то их двоичное представление — A=0000000000001110 и B=0000000000011000. В результате логического умножения A and B получим 0000000000001000 или 8 в десятичной системе счисления (рис. 2.2). Таким образом, A&B=14&24=8.

Рис. 2.2. Пример логического умножения

Арифметическое ИЛИ (|). Здесь также оба операнда переводятся в двоичную систему, после чего над ними происходит логическое поразрядное сложение операндов по следующим правилам:

1|1=1, 1|0=1, 0|1=1, 0|0=0.

Например, результат логического сложения чисел A=14 и B=24 будет равен A or B=30 (рис. 2.3).

Арифметическое исключающее ИЛИ (^). Оба операнда переводятся в двоичную систему, после чего над ними происходит логическая поразрядная операция ^ по следующим правилам:

1^1=0, 1^0=1, 0^1=1, 0^0=0.

Рис. 2.3. Пример логического сложения

Арифметическое отрицание (~). Эта операция выполняется над одним операндом. Применение операции ~ вызывает побитную инверсию двоичного представления числа (рис. 2.4).

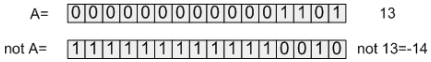

Рис. 2.4. Пример арифметического отрицания

Сдвиг влево (`M<<L`). Число `M`, представленное в двоичной системе, сдвигается влево на `L` позиций. Рассмотрим операцию `15 shl 3`. Число 15 в двоичной системе имеет вид `1111`. При сдвиге его на 3 позиции влево получим `1111000`. В десятичной системе это двоичное число равно `120`. Итак, `15 shl 3 =120` (рис. 2.5). Заметим, что сдвиг на один разряд влево соответствует умножению на два, на два разряда — умножению на четыре, на три — умножению на восемь. Таким образом, операция `M<<L` эквивалентна умножению числа `M` на 2 в степени `L`.

Рис. 2.5. Пример операции «Сдвиг влево»

Сдвиг вправо (`M>>L`). Число `M`, представленное в двоичной системе, сдвигается вправо на `L` позиций, что эквивалентно целочисленному делению числа `M` на 2 в степени `L`. Например, `15 shr 1=7` (рис. 2.6), `15 shr 3= 2`.

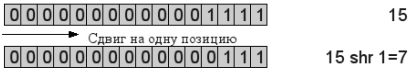

Рис. 2.6. Пример операции «Сдвиг вправо»

2.6.3 Логические операции

В C++ определены следующие логические операции || (или), && (и), ! (не). Логические операции выполняются над логическими значениями `true` (истина) и `false` (ложь). В языке C++ ложь — 0, истина — любое значение $\neq 0$. В таблице 2.6 приведены результаты логических операций.

Таблица 2.6: Логические операции

A	B	!A	A&&B	A||B
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

2.6.4 Операции отношения

Операции отношения возвращают в качестве результата логическое значение. Таких операций шесть: >, >=, <, <=, ==, !=. Результат операции отношения — логическое значение `true` (истина) или `false` (ложь).

2.6.5 Условная операция

Для организации разветвлений в простейшем случае можно использовать *условную операцию* ? :. Эта операция имеет три операнда и в общем виде может быть представлена так:

```
условие ? выражение1 : выражение2;
```

Работает операция следующим образом. Если условие истинно (не равно 0), то результатом будет `выражение1`, в противном случае `выражение2`. Например, операция `y=x<0 ? -x : x;` записывает в переменную `y` модуль числа `x`.

2.6.6 Операция преобразования типа

Для приведения выражения к другому типу данных в C++ существует *операция преобразования типа*:

`(тип) выражение;`

Например, в результате действий `x=5; y=x/2; z=(float) x/2;` переменная `y` примет значение равное 2 (результат целочисленного деления), а переменная `z = 2.5`.

2.6.7 Операция определения размера

Вычислить размер объекта или типа в байтах можно с помощью *операции определения размера*, которая имеет две формы записи:

`sizeof (тип);` или `sizeof выражение;`

Например, предположим, что была описана целочисленная переменная `int k=3;`. Исходя из того, что тип `int` занимает в памяти 4 байта, в переменную `m=sizeof k;` будет записано число 4.

В результате работы команд `double z=123.456; p=sizeof (k+z);` значение переменной `p` стало равно 8, т. к. вещественный тип `double` более длинный (8 байтов) по сравнению с типом `int` (4 байта) и значение результата было преобразовано к более длинному типу. В записи операции `sizeof (k+z)` были использованы скобки. Это связано с тем, что операция определения типа имеет более высокий приоритет, чем операция сложения. При заданном значении `z=123.456`; та же команда, но без скобок `p=sizeof k+z;` вычислит `p=4+123.456=127.456`.

Команда `s = sizeof "Hello"`; определит, что под заданную строку в памяти было выделено `s=6` байтов, т. к. объект состоит из 5 символов и один байт на символ окончания строки.

2.6.8 Операции с указателями

При работе с указателями часто используют операции *получения адреса* `&` и *разадресации* `*` (табл. 2.7).

Таблица 2.7: Операции получения адреса `&` и разадресации `*`

Описание	Адрес	Значение, хранящееся по адресу
`тип *p`	`p`	`*p`
`тип p`	`&p`	`p`

Операция получения адреса & возвращает адрес своего операнда. Например:

```
float a;     //Объявлена вещественная переменная a
float *adr_a; //Объявлен указатель на тип float
adr_a=&a;    //Оператор записывает в переменную adr_a
             //адрес переменной a
```

Операция разадресации * возвращает значение переменной, хранящееся по заданному адресу, т.е. выполняет действие, обратное операции &:

```
float a;     //Объявлена вещественная переменная a.
float *adr_a;   //Объявлен указатель на тип float.
a=*adr_a;     //Оператор записывает в переменную a
     //вещественное значение, хранящееся по адресу adr_a.
```

К указателям применяют *операцию присваивания*. Это значит, что значение одного указателя можно присвоить другому. Если указатели одного типа, то для этого применяют обычную операцию *присваивания*:

```
//Описана вещественная переменная и два указателя.
float PI=3.14159,*p1,*p2;
//В указатели p1 и p2 записывается адрес переменной PI.
p1=p2=&PI;
```

Если указатели ссылаются на различные типы, то при присваивании значения одного указателя другому, необходимо использовать преобразование типов. Без преобразования можно присваивать любому указателю указатель `void*`.

Рассмотрим пример работы с указателями различных типов:

```
float PI=3.14159;   //Объявлена вещественная переменная.
float *p1;     //Объявлен указатель на float.
double *p2;    //Объявлен указатель на double.
p1=&PI; //Переменной p1 присваивается значение адреса PI.
p2=(double *)p1;   //Указателю на double присваивается
            //значение, которое ссылается на тип float.
```

В указателях `p1` и `p2` хранится один и тот же адрес (`p1=0012FF7C`), но значения, на которые они ссылаются разные (`*p1=3.14159`, `*p2=2.642140e-308`). Это связано с тем, указатель типа `*float` адресует 4 байта, а указатель `*double` — 8 байт. После присваивания `p2=(double *)p1;` при обращении к `*p2` происходит следующее: к переменной, хранящейся по адресу `p1`, дописывается еще следующих 4 байта из памяти. В результате значение `*p2` не совпадает со значением `*p1`.

Таким образом, при преобразовании указателей разного типа приведение типов разрешает только синтаксическую проблему присваивания. Следует помнить, что операция * над указателями различного типа, ссылающимися на один и тот же адрес, возвращает различные значения.

Над адресами C++ определены следующие *арифметические операции*:

- *сложение и вычитание указателей с константой*;
- *вычитание одного указателя из другого*;
- *инкремент*;
- *декремент*.

Сложение и вычитание указателей с константой n означает, что указатель перемещается по ячейкам памяти на столько байт, сколько занимает n переменных того типа на который он указывает. Например, пусть указатель имеет символьный тип и его значение равно 100. Результат сложения этого указателя с единицей — 101, так как для хранения переменной типа char требуется один байт. Если же значение указателя равно 100, но он имеет целочисленный тип, то результат его сложения с единицей будет составлять 104, так как для переменной типа int отводится четыре байта.

Эти операции применимы только к указателям одного типа и имеют смысл в основном при работе со структурными типами данных, например массивами.

Фактически получается, что значение указателя изменяется на величину sizeof(тип). Если указатель на определенный тип увеличивается или уменьшается на константу, то его значение изменяется на величину этой константы, умноженную на размер объекта данного типа. Например:

```
//Объявление массива из 10 элементов.
double mas[10]={1.29,3.23,7.98,5.54,8.32,2.48,7.1};
double *p1;   //Объявление указателя на double
p1=&mas[0];   //Присвоение указателю адреса
        //нулевого элемента массива.
p1=p1+3;    //Увеличение значения адреса на 3*8=24
        //(размер типа double), в результате указатель
      //сместится на три ячейки, размером double каждая.
```

Вычитание двух указателей определяет сколько переменных данного типа размещается между указанными ячейками. Разность двух

указателей это разность их значений, деленная на размер типа в байтах. Так разность указателей на третий и нулевой элементы массива равна трем, а на третий и девятый — шести. Суммирование двух указателей не допускается.

Операции *инкремента* и *декремента*, соответственно, увеличивают или уменьшают значение адреса:

```
double *p1;
float *p2;
int *i;
p1++;   //Увеличение значения адреса на 8.
p2++;   //Увеличение значения адреса на 4.
i++;     //Увеличение значения адреса на 4.
```

К указателям так же применимы операции отношения `==`, `!=`, `<`, `>`, `<=`, `>=`. Иными словами указатели можно сравнивать. Например, если `i` указывает на пятый элемент массива, а `j` на первый, то отношение `i>j` истинно. Кроме того, любой указатель всегда можно сравнить на равенство с константой нулевого указателя (NULL)[6]. Однако, все эти утверждения верны, если речь идёт об указателях ссылающихся на один массив. В противном случае результат арифметических операций и операций отношения будет не определён.

2.7 Стандартные функции

В C++ определены *стандартные функции* над арифметическими операндами[7]. В таблице 2.8 приведены некоторые из них.

Таблица 2.8: Стандартные математические функции

Обозначение	**Действие**
`abs(x)`	Модуль целого числа x
`fabs(x)`	Модуль вещественного числа x
`sin(x)`	Синус числа x
`cos(x)`	Косинус числа x
`tan(x)`	Тангенс числа x
`atan(x)`	Арктангенс числа x, $x \in (-\frac{i\pi}{2}; \frac{i\pi}{2})$

[6]Константу нулевого указателя можно присвоить любому указателю и такой указатель при сравнении не будет равен любому реальному указателю.

[7]Работа с математическими функциями возможно только при подключении директивы math.h (п. **??**)

Таблица 2.8 — продолжение

Обозначение	**Действие**
`acos(x)`	Арккосинус числа x
`asin(x)`	Арксинус числа x
`exp(x)`	Экспонента, e^x
`log(x)`	Натуральный логарифм, $(x > 0)$
`log10(x)`	Десятичный логарифм, $(x > 0)$
`sqrt(x)`	Корень квадратный, $(x > 0)$
`pow(x,y)`	Возведение числа x в степень y
`ceil(x)`	Округление числа x до ближайшего большего целого
`floor(x)`	Округление числа x до ближайшего меньшего целого

Примеры записи математических выражений с использованием встроенных функций представлены в таблице 2.9.

Таблица 2.9: Примеры записи математических выражений

Математическая запись	**Запись на языке C++**
$\sqrt[3]{(a+b)^2}$	`pow((a+b)*(a+b),1./3)` или `pow(pow(a+b,2),1./3)`
$\cos^4(x)$	`pow(cos(x), 4)`
e^{2x}	`exp(2*x)`
$e^{5\sin(\frac{x}{2})}$	`exp(5*sin(x/2))`
$\sin^2(\sqrt{x})$	`pow(sin(sqrt(x)),2)`
$\ln(\lvert x-2\rvert)$	`log(fabs(x-2))`
$\log_b a$	`log(a)/log(b)`
$\frac{lg(x^2+1)}{lg(4)}$	`log10(x*x+1)/log10(4)`
$\sin(x^2+y^2)+\cos\frac{(x^2+y^2)}{2\cdot y}+$ $+\sqrt{x^2+y^2}$	`z=x*x+y*y;` `sin(z)+cos(z/(2*y))+sqrt(z);`

Определенную проблему представляет применение функции `pow(x,y)`. При программировании выражений, содержащих возведение в степень, надо внимательно проанализировать значения, которые могут принимать x и y, так как в некоторых случаях возведение x в степень y невыполнимо.

Так, ошибка возникает, если x — отрицательное число, а y — дробь. Предположим, что y — правильная дробь вида $\frac{k}{m}$. Если знаменатель m четный, это означает вычисление корня четной степени из отрицательного числа, а значит, операция не может быть выполнена. В противном случае, если знаменатель m нечетный, можно воспользоваться выражением `z = -pow(fabs(x),y)`. Например, вычисление кубического корня из вещественного числа можно представить командой:

```
z=(x<0) ? -pow(fabs(x),(double) 1/3) : pow(x,(double) 1/3);
```

2.8 Структура программы

Программа на языке C++ состоит из *функций*, *описаний* и *директив процессора*.

Одна из функций должна обязательно носить имя `main`. Элементарное описание функции имеет вид:

```
тип_результата имя_функции (параметры)
{
    оператор1;
    оператор2;
    ...
    операторN;
}
```

Здесь, `тип_результата` — это тип того значения, которое функция должна вычислить (если функция не должна возвращать значение, указывается тип `void`), `имя_функции` — имя, с которым можно обращаться к этой функции, `параметры` — список аргументов функции (может отсутствовать), `оператор1`, `оператор2`,..., `операторN` — операторы, представляющие тело функции, они обязательно заключаются в фигурные скобки и каждый оператор заканчивается точкой с запятой. Как правило программа на C++ состоит из одной или нескольких, не вложенных друг в друга функций.

Основному тексту программы предшествуют *директивы препроцессора* предназначенные для *подключения библиотек*, которые в общем виде выглядят так:

```
#include <имя_файла>
```

Каждая такая строка дает компилятору команду присоединить программный код, который хранится в отдельном файле с расширением `.h`. Такие файлы называют *файлами заголовков*. С их помощью можно выполнять ввод-вывод данных, работать с математическими

функциями, преобразовывать данные, распределять память и многое другое. Например, описание стандартных математических функций находится в заголовочном файле `math.h`.

Общую структуру программы на языке C++ можно записать следующим образом:

```
директивы процессора
описание глобальных переменных
тип_результата main(параметры)
{
описание переменных главной функции;
  операторы главной функции;
}

тип_результата имя1(параметры1)
{
описание переменных функции имя1;
  операторы1;
}

тип_результата имя2(параметры2)
{
описание переменных функции имя2;
  операторы2;
}

. . . . . .

тип_результата имяN(параметрыN)
{
описание переменных функции имяN;
  операторыN;
}
```

По месту объявления переменные в языке Си можно разделить на три класса: локальные, глобальные и формальные параметры функции.

Локальные переменные объявляются внутри функции и доступны только в ней. Например:

```
int main()
{
//В функции main определена вещественная переменная s,
    float s;
    s=4.5;    //и ей присвоено значение 4.5.
}
int f1()
{
//В функции f1 описана другая переменная s,
```

```
    int s;
    s=6;    //ей присвоено значение 6.
}
int f2()
{
//В функции f2 определена еще одна переменная s,
        long int s;
        s=25;        //ей присвоено значение 25.
}
```

Глобальные переменные описываются до всех функций и доступны из любого места программы. Например:

```
float s; //Определена глобальная переменная s.
int main()
{
//В главной функции переменной s присваивается значение 4.5.
        s=4.5;
}
int f1()
{
//В функции f1 переменной s присваивается значение 6.
        s=6;
}
int f2()
{
//В функции f2 переменной s присваивается значение 2.1.
    s=2.1;
```

Формальные параметры функций описываются в списке параметров функции. Работа с функциями подробно описана в главе **??**.

2.9 Ввод и вывод данных

Ввод-вывод данных в языке C++ осуществляется либо с помощью функций ввода-вывода в стиле C, либо с использованием библиотеки классов C++. Преимущество объектов C++ в том, что они легче в использовании, особенно если ввод-вывод достаточно простой. Функции ввода-вывода унаследованные от C более громоздкие, но более гибко управляют форматированным выводом данных.

Функция

`printf(строка форматов, список выводимых переменных);`

выполняет форматированный *вывод переменных*, указанных в списке, в соответствии со строкой форматов.

Функция

`scanf(строка форматов, список адресов вводимых переменных);`

выполняет *ввод переменных*, адреса которых указанны в списке, в соответствии со строкой форматов.

Строка форматов содержит символы, которые будут выводиться на экран или запрашиваться с клавиатуры и так называемые спецификации. *Спецификации* это строки, которые начинаются символом `%` и выполняют управление форматированием:

```
% флаг ширина . точность модификатор тип
```

Параметры `флаг`, `ширина`, `точность` и `модификатор` в спецификациях могут отсутствовать. Значения параметров спецификаций приведены в таблице 2.10.

Таблица 2.10: Символы управления

Параметр	**Назначение**
Флаги	
`-`	Выравнивание числа влево. Правая сторона дополняется пробелами. По умолчанию выравнивание вправо.
`+`	Перед числом выводится знак «+» или «-»
`Пробел`	Перед положительным числом выводится пробел, перед отрицательным «-»
`#`	Выводится код системы счисления: `0` — перед восьмеричным числом, `0x` (`0X`) перед шестнадцатеричным числом.
Ширина	
`n`	Ширина поля вывода. Если `n` позиций недостаточно, то поле вывода расширяется до минимально необходимого. Незаполненные позиции заполняются пробелами.
`0n`	То же, что и `n`, но незаполненные позиции заполняются нулями.
Точность	
`ничего`	Точность по умолчанию
`n`	Для типов `e`, `E`, `f` выводить `n` знаков после десятичной точки
Модификатор	
`h`	Для `d`, `i`, `o`, `u`, `x`, `X` тип `short int`.
`l`	Для `d`, `i`, `o`, `u`, `x`, `X` тип `long int`.
Тип	

Таблица 2.10 — продолжение

Параметр	Назначение
c	При вводе символьный тип char, при выводе один байт.
d	Десятичное int со знаком.
i	Десятичное int со знаком.
o	Восьмеричное int unsigned.
u	Десятичное int unsigned.
x, X	Шестнадцатеричное int unsigned, при x используются символы a-f, при X — A - F.
f	Значение со знаком вида [-]dddd.dddd.
e	Значение со знаком вида [-]d.dddde[+|-]ddd.
E	Значение со знаком вида [-]d.ddddE[+|-]ddd.
g	Значение со знаком типа e или f в зависимости от значения и точности.
G	Значение со знаком типа e или F в зависимости от значения и точности.
s	Строка символов.

Кроме того, строка форматов может содержать некоторые специальные символы, которые приведены в таблице 2.11.

Таблица 2.11: Специальные символы

Символ	Назначение
\b	Сдвиг текущей позиции влево.
\n	Перевод строки.
\r	Перевод в начало строки, не переходя на новую строку.
\t	Горизонтальная табуляция.
\’	Символ одинарной кавычки.
\’’	Символ двойной кавычки.
\?	Символ ?

Первой строкой программы, в которой будут применяться функции ввода-вывода языка C, должна быть директива #include <stdio.h>. Заголовочный файл stdio.h содержит описание функций ввода-вывода.

Рассмотрим работу функций на примере следующей задачи.

Задача 2.2. Зная a, b, c — длины сторон треугольника, вычислить площадь S и периметр P этого треугольника.

Входные данные: a, b, c. Выходные данные: S, P.

Для вычисления площади применим формулу Герона:

$S = \sqrt{r \cdot (r-a) \cdot (r-b) \cdot (r-c)}$, где $r = \frac{a+b+c}{2}$ — полупериметр.

Далее приведены две программы для решения данной задачи и результаты их работы (рис. 2.7–2.8). Сравните работу функций printf и scanf в этих программах.

```
//ЗАДАЧА 2.2 Вариант первый
#include <iostream>
#include <stdio.h>
#include <math.h>
using namespace std;
int main()
{
  float a,b,c,S,r;   //Описание переменных.
  printf("a=");      //Вывод на экран символов a=.
  //В функции scanf для вычисления адреса
  //переменной применяется операция &.
  scanf("%f",&a);    //Запись в переменную a значения
                     //введенного с клавиатуры.
  printf("b=");      //Вывод на экран символов b=.
  scanf("%f",&b);    //Запись в переменную b значения
                     //введенного с клавиатуры.
  printf("c=");      //Вывод на экран символов c=
  scanf("%f",&c);    //Запись в переменную с значения
                     //введенного с клавиатуры.
  r=(a+b+c)/2;       //Вычисление полупериметра.
  S=sqrt(r*(r-a)*(r-b)*(r-c)); //Вычисление площади треугольника.
  printf("S=%5.2f \t",S); //Вывод символов S=,
                          //значения S и символа
                          //табуляции \t.
                          //Спецификация %5.2f
                          //означает, что будет
                          //выведено вещественное
                          //число из пяти знаков,
                          //два из которых после точки.
  printf("p=%5.2f \n",2*r); //Вывод символов p=,
                            //значения выражения 2*r
                            //и символа окончания строки.
  //Оператор printf("S=%5.2f \t p=%5.2f \n",S,2*r);
  //выдаст тот же результат.
  return 0;
}

//ЗАДАЧА 2.2. Вариант второй
#include <iostream>
```

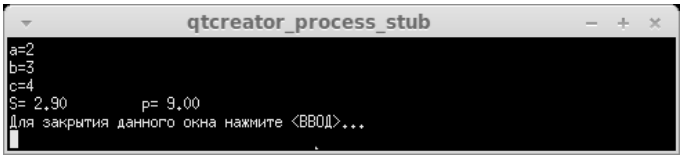

Рис. 2.7. Результаты работы программы к задаче 2.2 (вариант 1)

```
#include <stdio.h>
#include <math.h>
using namespace std;
int main()
{
  float a,b,c,S,r;
  printf("Vvedite a, b, c \n");   //Вывод на экран строки символов.
  scanf("%f%f%f",&a,&b,&c); //Ввод значений.
  r=(a+b+c)/2;
  S=sqrt(r*(r-a)*(r-b)*(r-c));
  printf("S=%5.2f \t p=%5.2f \n",S,2*r); //Вывод результатов.
  return 0;
}
```

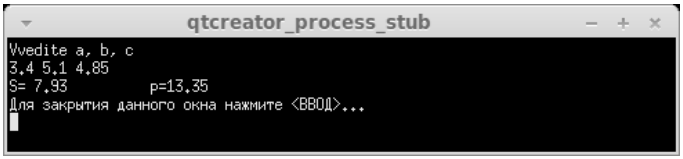

Рис. 2.8. Результаты работы программы к задаче 2.2 (вариант 2)

2.9.1 Объектно-ориентированные средства ввода-вывода.

Описание объектов для управления вводом-выводом содержится в заголовочном файле `iostream`. При подключении этого файла с помощью директивы `#include <iostream>` в программе автоматически создаются *объекты-потоки*[8] `cin` для *ввода с клавиатуры* и `cout` для

[8]Поток — виртуальный канал связи, создаваемый в программе для передачи данных

вывода на экран, а так же операции помещения в поток `<<` и чтения из потока `>>`.

Итак, с помощью объекта `cin` и операции `>>` можно ввести значение любой переменной. Например, если переменная `i` описана как целочисленная, то команда `cin>> i;` означает, что в переменную `i` будет записано некое целое число, введенное с клавиатуры. Если нужно ввести несколько переменных, следует написать `cin>>x>>y>>z;`.

Объект `cout` и операция `<<` позволяют вывести на экран значение любой переменной или текст. Текст необходимо заключать в двойные кавычки, кроме того, допустимо применение специальных символов `\t` и `\n` (таблица 2.11). Запись `cout<<i;` означает вывод на экран значения переменной `i`. А команда `cout<<x<<"\t"<<y;` выведет на экран значения переменных `x` и `y` разделенные символом табуляции.

Задача 2.3. Дано трехзначное число. Записать его цифры в обратном порядке и вывести на экран новое число.

Разберем решение данной задачи на конкретном примере. Здесь будут использоваться операции целочисленной арифметики.

Пусть `P=456`. Вычисление остатка от деления числа `P` на `10` даст его последнюю цифру (количество единиц в числе `P`): `456 % 10 =6.`

Операция деления нацело числа `P` на `10` позволит уменьшить количество разрядов и число станет двузначным:

`456 / 10 = 45.`

Остаток от деления полученного числа на `10` будет следующей цифрой числа `P` (количество десятков в числе `P`):

`45 % 10 = 5.`

Последнюю цифру числа `P` (количество сотен) можно найти так:

`456 / 100 = 4.`

Так как в задаче требовалось записать цифры числа `P` в обратном порядке, значит в новом числе будет `6` сотен, `5` десятков и `4` единицы:

`S = 6*100 + 5*10 + 4 = 654.`

Далее приведен текст программы, реализующей данную задачу для любого трехзначного числа.

```
#include <iostream>
using namespace std;
int main(int argc, char *argv[])
{
unsigned int P, S;   //Определение целочисленных
                     //переменных без знака.
cout<<"P=";          //Вывод на экран символов P=.
cin>>P;              //Ввод заданного числа P.
S=P%10*100+P/10%10*10+P/100;   //Вычисление нового числа S.
```

```
cout<<"S="<<S<<endl;                    //Вывод на экран символов S=
                                        //и значения переменной S.
return 0;
}
```

Задача 2.4. Пусть целочисленная переменная `i` и вещественная переменная `d` вводятся с клавиатуры. Определить размер памяти, отведенной для хранения этих переменных и их суммы, в байтах. Вычислить сколько памяти будет выделено для хранения строки `C Новым Годом!`. Вывести на экран размеры различных типов данных языка `C++` в байтах.

Далее приведён текст программы.

```
#include <iostream>
using namespace std;
int main()
{
int i;      //Определение целочисленной переменной.
double d;   //Определение вещественной переменной.

cout<<"i= "; cin>>i;   //Ввод переменной i.
cout<<"d= "; cin>>d;   //Ввод переменной d.
//Размер памяти, отведенной под переменную i.
cout<<"Размер i: "<<sizeof i<<"\n";
//Размер памяти, отведенной под переменную d.
cout<<"Размер d: "<<sizeof d<<"\n";
//Размер памяти, отведенной под значение выражения i+d.
cout<<"Размер i+d: "<<sizeof (i+d)<<"\n";
cout<<"Размер строки <C Новым Годом!>:";
//Размер памяти, отведенной под строку.
cout<<sizeof "C Новым годом!"<<"\n";
//Вычисление размеров различных типов данных:
cout<<"Размер char: "<<sizeof (char)<<"\n";
cout<<"Размер int: "<<sizeof (int)<<"\n";
cout<<"Размер short int: "<<sizeof (short int)<<"\n";
cout<<"Размер long int: "<<sizeof (long int)<<"\n";
cout<<"Размер long long int:";
cout<<sizeof (long long int)<<"\n";
cout<<"Размер float: "<<sizeof (float)<<"\n";
cout<<"Размер double: "<<sizeof (double)<<"\n";
cout<<"Размер long double: "<<sizeof (long double)<<"\n";
return 0;
}
```

Результаты работы программы[9]

[9]Обратите внимание, что при использовании кодировки utf-16 один кириллический символ занимает в памяти занимает 2 байта.

```
i= 23
d= 45.76
Размер i: 4
Размер d: 8
Размер i+d: 8
Размер <С Новым годом!>:26
Размер char: 1
Размер int: 4
Размер short int: 2
Размер long int: 4
Размер long long int:8
Размер float: 4
Размер double: 8
Размер long double: 12
```

2.10 Задачи для самостоятельного решения

2.10.1 Ввод-вывод данных. Операция присваивания.

Разработать программу на языке C++. Все входные и выходные данные в задачах — *вещественные числа*. Для ввода и вывода данных использовать функции `scanf` и `printf`.

1. Даны катеты прямоугольного треугольника a и b. Найти гипотенузу c и углы треугольника α, β, χ.
2. Известна гипотенуза c и прилежащий угол α прямоугольного треугольника. Найти площадь треугольника S и угол β.
3. Известна диагональ квадрата d. Вычислить площадь S и периметр P квадрата.
4. Дан диаметр окружности d. Найти ее длину L и площадь круга S.
5. Даны три числа — a, b, c. Найти их среднее арифметическое и среднее геометрическое.
6. Даны катеты прямоугольного треугольника a и b. Найти его гипотенузу c и периметр P.
7. Дан длина окружности L. Найти ее радиус R и площадь круга S.
8. Даны два ненулевых числа a и b. Найти сумму S, разность R, произведение P и частное d их квадратов.
9. Поменять местами содержимое переменных A и B и вывести новые значения A и B.
10. Точки A и B заданы координатами на плоскости: $A(x_1, y_1)$, $B(x_2, y_2)$. Найти длину отрезка AB.

11. Заданы два катета прямоугольного треугольника a и b. Вычислить его площадь S и периметр P.
12. Даны переменные A, B, C. Изменить их значения, переместив содержимое A в B, B — в C, C — в A, и вывести новые значения переменных A, B, C.
13. Известна диагональ ромба d. Вычислить его площадь S и периметр P.
14. Найти значение функции $y = 4 \cdot (x + 1)^3 + 5 \cdot (x - 1)^5 + 2$ и ее производной при заданном значении x.
15. Даны два ненулевых числа a и b. Найти сумму S, разность R, произведение P и частное D их модулей.
16. Известны координаты вершин квадрата $ABCD$: $A(x_1, y_1)$ и $C(x_2, y_2)$. Найти его площадь S и периметр P.
17. Даны длины сторон прямоугольника a и b. Найти его площадь S и периметр P.
18. Известно значение периметра P равностороннего треугольника. Вычислить его площадь S.
19. Задан периметр квадрата P. Вычислить сторону квадрата a, диагональ d и площадь S.
20. Дана сторона квадрата a. Вычислить периметр квадрата P, его площадь S и длину диагонали d.
21. Три точки заданы координатами на плоскости: $A(x_1, y_1)$, $B(x_2, y_2)$ и $C(x_3, y_3)$. Найти длины отрезков AB и BC.
22. Даны переменные A, B, C. Изменить их значения, переместив содержимое A в C, C — в B, B — в A, и вывести новые значения переменных A, B, C.
23. Даны числа — a_1, a_2, a_3, a_4, a_5. Найти их среднее арифметическое и среднее геометрическое значения.
24. Найти значение функции $y = \frac{3}{2} \cdot (x + 3)^4 - \frac{1}{5} \cdot (x - 1)^5$ и ее производной при заданном значении x.
25. Точки A и B заданы координатами в пространстве: $A(x_1, y_1, z_1)$, $B(x_2, y_2, z_2)$. Найти длину отрезка AB.

2.10.2 Операции целочисленной арифметики.

Разработать программу на языке C++. Все входные данные в задачах — целые числа. Для ввода и вывода данных использовать объектно-ориентированные средства ввода-вывода.

1. Расстояние L задано в сантиметрах. Найти количество полных метров в нем и остаток в сантиметрах.

2. Масса M задана в килограммах. Найти количество полных тонн в ней и остаток в килограммах.
3. Дан размер файла Bв байтах. Найти количество полных килобайтов, которые занимает данный файл и остаток в байтах.
4. Дано двузначное число. Вывести на экран количество десятков и единиц в нем.
5. Дано двузначное число. Найти сумму его цифр.
6. Дано двузначное число. Найти произведение его цифр.
7. Дано двузначное число. Вывести число, полученное при перестановке цифр исходного числа.
8. Дано трехзначное число. Определить сколько в нем единиц, десятков и сотен.
9. Дано трехзначное число. Найти сумму его цифр.
10. Дано трехзначное число. Найти произведение его цифр.
11. Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и десятков исходного числа.
12. Дано трехзначное число. Вывести число, полученное при перестановке цифр сотен и единиц исходного числа.
13. Дано трехзначное число. Вывести число, полученное при перестановке цифр десятков и единиц исходного числа.
14. С начала суток прошло N секунд. Найти количество полных минут, прошедших с начала суток и остаток в секндах.
15. С начала суток прошло N секунд. Найти количество полных часов, прошедших с начала суток и остаток в секндах.
16. Дано двузначное число $\alpha \leq 88$. Вывести на экран число, которое получится если каждую цифру числа a увеличить на единицу.
17. Дано двузначное число $\alpha \geq 22$. Вывести на экран число, которое получится если каждую цифру числа a уменьшить на единицу.
18. Расстояние L задано в метрах. Найти количество полных километров в нем и остаток в метрах.
19. Масса M задана в граммах. Найти количество полных килограммов в ней и остаток в граммах.
20. Размер файла B дан в килобайтах. Найти количество полных мегабайтов, которые занимает данный файл и остаток в килобайтах.
21. Расстояние L задано в дециметрах. Найти количество полных метров в нем и остаток в сантиметрах.
22. С начала года прошло K дней. Найти количество полных недель, прошедших с начала года и осток в днях.
23. С начала года прошло K часов. Найти количество полных дней, прошедших с начала года и осток в часах.

24. Дано двузначное число $a \leq 44$. Вывести на экран число, которое получится если удвоить каждую цифру числа a.
25. Дано двузначное число $\alpha \geq 22$. Вывести на экран число, которое получится если каждую цифру числа a уменьшить вдвое.

2.10.3 Встроенные математические функции

Разработать программу на языке C++. Все входные и выходные данные в задачах — *вещественные числа*. Для ввода и вывода данных использовать функции `scanf` и `printf`.

Вычислить значение выражения $y = f(x)$ при заданном значении x. Варианты заданий представлены в таблице 2.12.

Таблица 2.12: Задачи для самостоятельного решения

№	**Выражение** $f(x)$
1	$\sqrt[7]{x^2 + 2.7 \cdot \pi \cdot \cos\sqrt{\lvert x^3\rvert} - 2} + e^x$
2	$\mathrm{tg}^4 x + \sin^2\frac{\pi}{x} - e^{2x^2+3.6x-1}$
3	$\left\lvert \lvert x^4 - \cos x\rvert - \sqrt[9]{1+\sqrt{x^6}} \right\rvert + \sin^3\frac{\pi}{e^x+1}$
4	$\log_4 \lvert e^x - 4\rvert - \sqrt[7]{\left\lvert \frac{2\cdot x}{3.21 + \cos^2\frac{\pi}{7}} \right\rvert}$
5	$\sqrt[3]{\sqrt{\lvert x\rvert}} + \lvert \mathrm{ctg}^2 x + \frac{e^x}{2\cdot\pi} - x^3\rvert$
6	$x^5 + \log_3^2(3x^2+5) + \sqrt[9]{(\pi - 6x^2)^2}$
7	$\frac{1 - \log\lvert x - \cos(2x-\pi)\rvert}{6 + x^{4x-1}} + \sqrt[5]{x^3}$
8	$e^{x+\frac{\pi}{3}} + \sqrt[3]{tg\left\lvert \frac{x^5}{x^2+13.22} \right\rvert} + \cos^3 x$
9	$x^{1+\frac{3\cdot\pi}{4}} - 3x^3 - \sqrt[5]{(x+1)^4 + \lg\left\lvert \frac{x}{x+1} \right\rvert}$
10	$\sqrt[5]{x^3 + \cos\sqrt{\lvert x^3\rvert}} + \frac{e^x}{\cos(3\cdot x + \frac{\pi}{15})}$
11	$e^{2x} + \sqrt[5]{ctg\frac{(x-\pi)^9}{x^4+3.4}} + \sin^2 6.2x$
12	$\sqrt[5]{(x+tga)^2} - \frac{1 - \ln\lvert e^x + \cos\frac{\pi}{8}\rvert}{2}$

Таблица 2.12 — продолжение

№	**Выражение** $f(x)$
13	$\log(e^x+27)-\sqrt{\left\lvert x^3+\dfrac{\sqrt[5]{x^7}+14}{\sin 5x+5.1\cdot\pi}\right\rvert}$
14	$\ln\lvert\cos(x-2\cdot\pi)\rvert-\sqrt[3]{1+\dfrac{e^x}{\sin x-3}}$
15	$\sqrt{\left\lvert x^3+\dfrac{\sqrt[3]{x^4}-1}{\sin x+\pi+e^x}\right\rvert}$
16	$\sqrt[3]{\dfrac{1+3\cdot\pi}{1+x^2}}+\lvert\operatorname{arctg}^2 x^3\rvert$
17	$\operatorname{tg}^2\lvert x\rvert+3^{2x^2-e^x}+\dfrac{\sqrt[7]{x^2}}{\cos^2\pi x}$
18	$x^4-\sqrt[5]{\pi-\sqrt{\lvert x^3\rvert}+\sin^2\dfrac{x}{x^2+1}}$
19	$\log(e^x+6)-\sqrt[3]{(x-4)^2+1.47\sin\sqrt{\lvert\pi\cdot x\rvert}}$
20	$\dfrac{x^5}{\sin\lvert x-7\rvert}+\log^2(x^2+2.5)-\sqrt[3]{(\pi-6.1x^2)^2}$
21	$\operatorname{ctg}^2\dfrac{x\cdot\pi}{3}-\left(\sqrt{\lvert x\rvert}-3.4\right)^{x^2-10}+\ln(x^2+3)$
22	$\left\lvert\log_5\lvert x^3-e^x\rvert-\sqrt[3]{\dfrac{2x}{\cos(x+1.23\cdot\pi)}}\right\rvert$
23	$\left\lvert\lvert\cos\dfrac{\pi}{7}-e^x\rvert-\sqrt[7]{2+\sqrt{x^5}}\right\rvert+\ln\dfrac{x^4+1}{6}$
24	$\log(x^2+2)-\sin^2 x+\sqrt[5]{2-\sqrt{\lvert x\rvert}+\sin\dfrac{\pi}{e^x+1}}$
25	$\log_2 e^x-\cos\dfrac{x}{\pi}+\sqrt[3]{\dfrac{\lvert tg(2x)\rvert}{2.6+x^2+x^3}}$

Предметный указатель

Научное издание

Серия «Библиотека ALT Linux»

Алексеев Евгений Ростиславович, Григорий Григорьевич Злобин,
Дмитрий Александрович Костюк, Чеснокова Оксана Витальевна,
Чмыхало Александр Сергеевич

Программирование на языке C++ в среде Qt Creator

Оформление обложки: А. С. Осмоловская
Вёрстка: В. Л. Черный
Редактура: В. Л. Черный

Подписано в печать 26.08.14. Формат 60x90/16.
Гарнитура Computer Modern. Печать офсетная. Бумага офсетная.
Усл. печ. л. 23,0. . Тираж 999 экз. Заказ

ООО «Альт Линукс»
Адрес для переписки: 119334, Москва, 5-й Донской проезд, д. 15, стр. 6
Телефон: (495) 662-38-83. E-mail: sales@altlinux.ru
`http://altlinux.ru`