# ЯЗЫК

# ПРОГРАММИРОВАНИЯ

# С++

Балашиха 2010

# 1. ОСНОВНЫЕ ЭЛЕМЕНТЫ ПРОГРАММЫ

## 1.1. Изучаемая версия языка

Существует большое количество версий языка Си. Мы будем изучать язык Турбо Си 2.0 фирмы Borland. При желании можно пользоваться версиями Турбо С++ 3.0 или Borland C++ 3.1. Для самостоятельной работы с этим языком программирования рекомендуется книга [3], в которой рассматривается версия языка Си фирмы Borland.

## 1.2. Программа «Hello World»

### 1.2.1. Пример 1.1

Основные элементы рассмотрим на примере простейшей программы, предложенной ее создателем Денисом Ритчи [1]. При вводе программы необходимо точно соблюдать ее синтаксис, а именно:

– вводить программу строчными буквами;

– некоторые строки программы заканчиваются символом точка с запятой.

***Листинг 1.1***

/\* Программа `Привет мир!` \*/

#include <stdio.h> /\* подключение файла библиотеки \*/

void main(void) /\* главная функция \*/

{ /\* начало блока \*/

printf(“Hello world!\n”); /\* вывод на экран сообщения \*/

} /\* конец блока \*/

Рассмотрим основные элементы программы.

### 1.2.2. Комментарии

Комментарий начинается с символов /\* и заканчивается теми же символами, идущими в обратном порядке \*/. Они служат для ввода пояснений в программе. Например, строка

/\* главная функция \*/

представляет собой комментарий. При запуске программы комментарии компилятором языка игнорируются.

Обычно принято начинать программу с комментария, в котором указано назначение программы. В комментарии можно использовать любые символы, в том числе и символы русского алфавита. Каждый комментарий может занимать строку программы целиком, часть ее или несколько строк. Например, допустим следующий вариант:

/\* Это

просто

строка

комментария \*/

### 1.2.3. Заголовочные файлы

Вначале содержательной части программы на языке Си всегда используются так называемые «директивы препроцессора», начинающиеся с символа #. В данной программе используется строка

#include <stdio.h>

которая означает, что необходимо подключить заголовочный файл «stdio.h». Это позволяет использовать в программе библиотеку стандартного ввода-вывода, а именно, в данной программе используется функция стандартного вывода данных на экран printf().

При использовании в программе функций из других библиотек для подключения последних необходимо включить соответствующий файл. В программах наиболее часто будут встречаться следующие заголовочные файлы:

|  |  |
| --- | --- |
| stdio.h | библиотека стандартного ввода-вывода, например, для использования функций ввода с клавиатуры и вывода на экран scanf() и printf(), ввода/вывода в файл fprintf()и fscanf() |
| conio.h | библиотека консольного ввода-вывода, например, для функций ввода одиночного символа getch(), конольного ввода/вывода cscanf() и cprintf(), очистки экрана clrscr(), перемещения курсора gotoxy(), изменения цвета символов textcolor() |
| math.h | математическая библиотека, например, для функций sqrt(), sin(), cos(), log(), exp() |
| graphics.h | графическая библиотека, например, для функций инициализации графики initgraph(), рисования графических примитивов line(), rectangle() |

При необходимости использования функций из нескольких библиотек в начале программы указываются все требуемые заголовочные файлы. Например, при использовании математических функций, функций стандартного и консольного ввода-вывода в заголовке программы записываются следующие строки:

#include <stdio.h>

#include <conio.h>

#include <math.h>

### 1.2.4. Главная функция

Листинг программы на Си содержит одну или несколько функций. Обязательным является использование главной функции – функции main(). Строка описания главной функции в приведенной программе содержит ключевое слово void. Это означает, что после окончания функции ничего не возвращается. В общем виде описание каждой функции, в том числе и функции main(), имеет следующий синтаксис (символы русского алфавита и знаки < и > используются только для пояснения):

<ТипВозвращаемогоЗначения> <ИмяФункции> (<СписокФормальныхПеременных>)

где <ТипВозвращаемогоЗначения> – результат исполнения функции, точнее тип возвращаемого значения переменной (допустимо использовать разрешенные типы, например, void, int, float и т.д.);

<ИмяФункции> – идентификатор (имя) функции;

<СписокФормальныхПеременных> – перечень типов и идентификаторов формальных переменных.

Чтобы стала понятной эта форма описания функции, приведем запись функции в естественной математической форме:

y = sin ( x )

ТипВозвращаемогоЗначения ИмяФункции СписокФормальныхПеременных

В рассматриваемом примере программы «Hello World!» строка описания заголовка функции

void main(void)

означает, что имя функции main, список формальных параметров отсутствует и после ее завершения ничего не возвращается.

### 1.2.5. Начало и конец функции или блока

Любая функция на языке содержит «тело» – список операторов, которые выполняются при ее вызове. Принадлежность операторов данной функции определяется символами открывающей фигурной скобки { и закрывающей скобки }. В общем случае эти символы начала и конца блока можно использовать не только для функций, но и для операторов управления, позволяющих изменить ход исполнения программы: операторов if, switch, for, while , do-while.

### 1.2.6. Вывод на экран сообщения

Для вывода данных на экран в языке Си имеется достаточно большое количество средств. Наиболее часто используется функция printf() из стандартной библиотеки ввода-вывода. В простейшем случае, как в приведенном примере, эта функция содержит только один параметр – строку, начинающуюся и завершающуюся символом двойной кавычки ″…″. Все, что заключено внутри строки, выводится на экран при исполнении программы. Содержимое строки может быть записано как на латинском, так и на русском языке.

Помимо обычного текста в строке можно указать специальные символы, позволяющие вывести значения переменных, выполнить перевод строки, горизонтальную табуляцию, подать звуковой сигнал и т.д. Специальный символ начинается с символа обратного слеша, например:

\n – перевод строки;

\t – горизонтальная табуляция (перенос курсора вправо на несколько позиций);

\а – подача звукового сигнала.

Количество и последовательность специальных символов не ограничивается. Например, вызов функции

printf(“Эта\n\tПросто\n\t\tСтрока\n”);

позволит вывести на экран сообщение в нескольких строках со смещением в каждой новой строке:

Эта

Просто

Строка

В рассматриваемом примере программы на экран выводится сообщение “Hello world!” и курсор переводится на новую строку.

Следует отметить еще одну важную деталь – практически все операторы внутри функции заканчиваются символами точка с запятой, операторы вне функций обычно записываются без этих символов.

### 1.2.7. Запуск программы на компиляцию, компоновку и исполнение

После записи в редакторе языка программы для ее запуска необходимо выполнить следующую последовательность действий:

1) нажать клавишу F2 и ввести имя файла – при этом выполняется сохранение файла на диске (расширение «\*.c» указывать необязательно);

2) нажать клавиши ALT-F9 – запустить программу на компиляцию, при этом проверяется синтаксис написания программы. После компиляции на экране появляется окно компилятора (рис. 1.1), содержащее информацию об ошибках и предупреждениях.

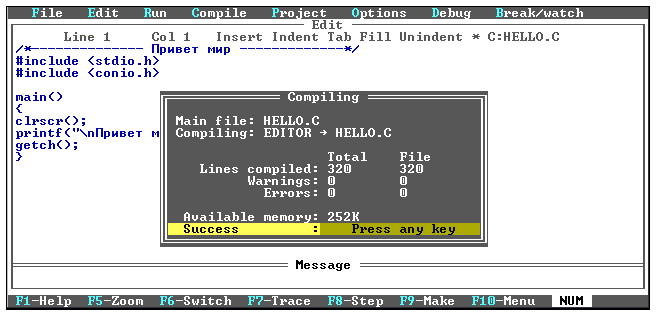


Рис. 1.1. Компиляция программы

Если программа написана без ошибок, то можно выполнять дальнейшие действия (в строках Errors и Warning окна компилятора должны стоять цифры 0);

3) нажать клавиши Ctrl-F9 – запустить компоновщик программы и при нулевом количестве ошибок программа запускается на исполнение (компоновщик подключает библиотечные файлы);

4) нажать клавиши Alt-F5 – окно просмотра исполнения программы.

После выполнения всех этих операций с рассматриваемой программой «Hello World!» на экране появится результат – выполнится вывод сообщения:

Hello world!

## 1.3. Программа ввода данных и вывода суммы

### 1.3.1. Пример 1.2

Усложним программу: необходимо ввести с клавиатуры два целых чисел и результат вывести на экран.

***Листинг 1.2***

/\* Расчет суммы двух чисел \*/

#include <stdio.h>

#include <conio.h>

main()

{

int a,b,c; /\* объявление трех целых переменных \*/

clrscr(); /\* очистка экрана \*/

printf(“Введите два целых числа:\n”);

scanf(“%d%d”,&a,&b); /\* ввод с клавиатуры двух переменных \*/

c=a+b; /\* выполнение арифметической операции \*/

printf(“Сумма чисел равна: %d\n”,c);

/\* вывод результата \*/

getch(); /\* ожидание нажатия любой клавиши \*/

}

Рассмотрим только новые элементы программы.

### 1.3.2. Подключение библиотеки консольного ввода – вывода выполняется строкой

#include <conio.h>

Эта библиотека необходима для использования двух новых функций: очистки экрана вначале программы

clrscr();

и ожидания нажатия любой клавиши после ее завершения

getch();

Кроме всего библиотека <conio.h> используется для ввода–вывода данных в указанное место на экране, с заданными атрибутами (цветом фона и символов).

### 1.3.3. Объявление трех переменных целого типа выполняется строкой

int a,b,c;

После этого в программе можно использовать переменные с именами a, b и c. Все переменные, которые будут использоваться в программе, написанной на языке Си, должны быть обязательно объявлены. Объявление переменных выполняется сразу после открывающей фигурной скобки и определяет имя переменной и её тип. К основным типам переменных относятся переменные типа символ (обозначаются ключевым словом ‘char’), целое число (ключевое слово ‘int’), типа вещественное число (ключевое слово ‘float’). В данном примере используются три целые переменные, поэтому применяется ключевое слово ‘int’, а сами переменные перечисляются через запятую. В конце строки обязательно используется точка с запятой, означающая конец объявления переменных.

### 1.3.4. Форматированный ввод значений переменных с клавиатуры

Выполняется функцией форматированного ввода scanf(). Первый параметр функции является строкой “%d%d”, содержащей, так называемые, спецификации формата ввода переменных. Спецификация начинается с символа %, за которым следует код формата, например, формат для целого числа представляет собой последовательность символов %d. В данном примере строка определяет ввод двух целых переменных. Остальные параметры функции scanf() определяют адреса вводимых переменных, т.е. каким переменным будут присвоены введенные значения. При указании адреса переменной на языке Си перед ее именем устанавливается знак & конъюнкции.

### 1.3.4. Вычисление значения переменной

Вычисление в выражении

c=a+b;

выполняется в два этапа: вначале находится сумма значений переменных а и b, затем результат присваивается переменной с. Операция суммирования двух переменных, также как и остальные операторы в теле функции main(), заканчивается точкой с запятой.

### 1.3.5. Форматированный вывод на экран

В строке “Сумма чисел равна: %d\n”, используемой в функции форматированного вывода printf() помимо простого сообщения содержится спецификация формата вывода на экран значения целой переменной. Однако, в отличие от функции scanf(), в функции printf() после строки идет перечень имен переменных, а не их адресов.

### 1.3.6. Исполнение программы

После ввода программы, ее компиляции, компоновки и запуска на исполнение на чистом экране появится сообщение “Введите два целых числа:”. После этого необходимо будет ввести два целых числа. Пример выполнения программы приведен ниже:

Введите два целых числа:

10

13

Сумма чисел равна: 23

## 1.4. Интегрированная среда разработки программ

### 1.4.1. Окно редактора

При работе с программой Турбо Си все средства разработки программы, а именно, редактирование, компиляция, компоновка, отладка программы, находятся под рукой. Эти средства называются интегрированной средой разработки (оболочкой программы). На рис. 1.2 представлена интегрированная среда с набранной программой «Привет мир!».

Строка меню позволяет выполнить команды интегрированной среды, такие как сохранение и загрузка файла, компиляция, создание файла проекта и т.д. В рабочей области выполняются операции редактирования программы, выводятся сообщения о компиляции, компоновке и отладчике. В верхней части рабочей области в режиме редактирования приводится строка состояния, содержащая координаты текущего положения курсора в файле (Line, Col – номера строки и столбца), режим вставки/замещения символов (Insert – режим вставки). В конце этой строки приводится имя редактируемого файла.

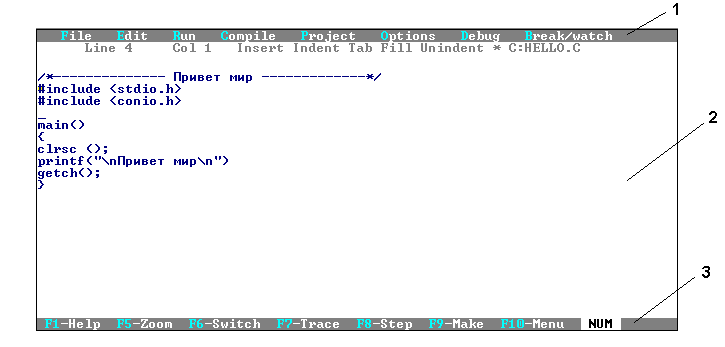


Рис. 1.2. Интегрированная среда разработки

### 1.4.2. Пункты меню интегрированной среды

Вход в меню среды разработки выполняется нажатием клавиши F10. Состав и назначение пунктов меню интегрированной среды разработки представлено в табл. 1.1.

Пункты меню интегрированной среды разработки

Таблица 1.1

|  |  |  |
| --- | --- | --- |
| Пункт меню | Подпункт / команда | Назначение |
| File |  | файловые команды |
|  | Load | загрузка файла с диска |
|  | New | создание нового файла |
|  | Save | сохранение файла на диске |
|  | Write to | сохранение файла под другим именем |
|  | Quit | выход из интегрированной среды |
| Edit |  | редактирования файла |
| Run |  | запуск на исполнение программы, а также инициализация и завершение режима отладки программы |
| Compile |  | компиляция программы |
| Project |  | команды управления проектом многофайловой программы |
| Options |  | команды просмотра и изменения параметров среды разработки |
|  | Compiler | параметры компилятора (модели памяти, коды генерации и т.д.) |
|  | Linker | параметры компоновщика (подключение графической библиотеки и др.) |
|  | Environment | параметры среды (автосохранение текущего редактируемого файла и файла конфигурации, режим экрана и др.) |
|  | Directories | указание на местоположение включаемых файлов, библиотечных файлов, файла помощи (справки) |
|  | Arguments | аргументы командной строки |
|  | Save options | сохранение параметров среды разработки |
| Debug |  | отладчик интегрированной среды |
| Break/watch |  | точки останова остановки программ и окно просмотра изменяющихся значений |

### 1.4.3. Основные команды среды разработки (горячие клавиши)

Приведены в табл. 1.2.

Таблица 1.2.

|  |  |
| --- | --- |
| Команда | Назначение |
| **Общие команды** |  |
| F10 | Вход в меню |
| ESC | Выход в редактор |
| Alt+F9 | Компиляция |
| ^F9 | Компоновка |
| Alt+F5 | Просмотр окна исполнения программы |
| F5 | Развернуть/свернуть текущее окно |
| F6 | Перейти к следующему окну |
| ^PgUp | Перейти к началу файла |
| ^PgDn | Перейти к концу файла |
| Alt+X | Выход из среды разработки |
| Команда | Назначение |
| **Файловые команды** |  |
| F2 | Сохранение файлов на диске |
| F3 | Загрузка файла с диска |
| Alt+F3 | Загрузка файла из списка последних 9 файлов |
| **Блочные команды** |  |
| ^K^B | Отметить начало блока |
| ^K^K | Отметить конец блока |
| ^K^C | Вставить блок |
| ^K^V | Перенести блок |
| ^K^Y | Вырезать блок |
| ^K^H | Отменить выделение блока |
| ^K^R | Вставить файл с диска |
| ^K^W | Записать блок на диск |
| ^K^P | Печатать блок на принтере |
| ^Y | Удалить строку |
| **Команды поиска** |  |
| ^Q^F | Поиск |
| ^Q^A | Поиск с заменой |
| ^L | Повторить последний поиск |
| ^Q^[ | Поиск парных фигурной скобки { }, квадратной [ ], круглой ( ), угловой < >, комментариев /\* \*/, двойной кавычки « », апострофа ‘ ’ |

Примечание: Символ ‘^’ обозначает клавишу ‘CTRL’

**1.5. Общие требования при написании и оформлении программ**

1.5.1. В программах обязательно должны быть следующие элементы:

– комментарии;

– название программы;

– очистка экрана в начале программы, ожидание нажатия клавиши в конце;

– при использовании операторов управления подчиненные операторы записывать со сдвигом вправо.

1.5.2. Отчет по каждой задаче должен содержать:

– исходное условие;

– листинг программы (сама программа на языке СИ);

– алгоритм исполнения программы (если размер исходного текста программы превышает одну страницу, то можно показывать на блок-схеме сгруппированные операторы, но операторы управления должны быть показаны все);

– экран исполнения программы (как мы увидим один из возможных вариантов исполнения программы на экране).

## 1.6. Задания для решения

1.6.1. Напишите программу, в которой на экран выводятся фамилия, имя, отчество, при этом каждое слово выводится с новой строки.

1.6.2. Измените предыдущую программу таким образом, чтобы текст выводился в центре экрана (в текстовом режиме экран содержит 25 строк, 80 столбцов).

1.6.3. Напишите программу возведения целого числа в квадрат: с клавиатуры вводится целое число, это число умножается само на себя и на экран выводится результат.

1.6.4. Напишите программу расчета суммарного сопротивления параллельно соединенных двух (или трех) сопротивлений (вводятся целые значения сопротивлений, рассчитывается и выдается на экран суммарное значение). Экран исполнения программы должен быть примерно такой:

Введите R1: \_\_

Введите R2: \_\_

Суммарное сопротивление R1||R2: \_\_

1.6.5. По введенным с клавиатуры значениям сопротивления R и приложенного напряжения U программа рассчитывает ток I, протекающий через сопротивление и мощность P, выделяемую на нем. Экран исполнения программы должен быть примерно такой:

Введите R= \_\_

Введите U= \_\_

Расчет тока и мощности:

Ток I= \_\_

Мощность P= \_\_

1.6.6. На основе листинга 1.2 составьте программу, которая выдает одновременно результат деления двух целых чисел (операция '/') и остаток от деления двух целых чисел (операция '%'). Выполните анализ полученной программы – выводится ли дробная часть числа, выполняется ли округление, делятся ли отрицательные числа, делятся ли вещественные числа?

1.6.7. Модифицируйте предыдущую программу для деления вещественных чисел (смените тип данных – вместо int установите float, и спецификации формата – вместо %d установите %f). Выполните анализ полученных при делении чисел – выводится ли дробная часть числа, делятся ли отрицательные числа, делятся ли целые числа?

1.6.8. Напишите программу для ввода комплексного числа в алгебраической форме: вещественной и мнимой частей, и вывода в показательной форме: модуля и аргумента комплексного числа.

*Рекомендации при составлении программы*. Для использования математических функций (гармонических, показательных, логарифмических и т.д.) в программы на языке Си включается файл <math.h>. Перечень всех математических функций и их описанием можно просмотреть используя контекстную помощь (нажать клавиши Ctrl+F1 на имени включаемого файла <math.h>). Для написания заданной программы необходимо: использовать функцию квадратного корня (функция sqrt). Для ввода вещественных чисел смените тип данных – вместо int установите float, и спецификации формата – вместо %d установите %f).

# 2. ПЕРЕМЕННЫЕ И КОНСТАНТЫ

Любая программа обрабатывает информацию или данные, в программе можно использовать переменные и константы. Как понятно из названия переменные во время исполнения программы могут изменять свои значения. Эти значения могут изменяться в определенном диапазоне в соответствии с заданным типом данных. Термин константа в общем случае относится к данным, которые не могут быть изменены во время исполнения программы.

## 2.1. Типы данных

В языке Си можно использовать различные типы данных для хранения и обработки информации. Данные каждого типа могут принимать значения в некотором заранее известном диапазоне. В памяти они занимают определенное количество байт. Таким образом, тип данных - это способ хранения информации в памяти.

В зависимости от реализации языка Си размер и диапазон изменения значений различных типов может меняться. В языке Турбо Си основными типами являются целые числа и вещественные (действительные) числа, иногда называемые числами с плавающей запятой. В таблице 2.1 приведены основные типы данных, доступных для языка ТС.

Основные типы данных языка ТС

Таблица 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных | Название типа | Размер занимаемой памяти, Байт | Диапазон допустимых значений |
| char | символ | 1 | -128... 127 |
| unsigned char | беззнаковый символ | 1 | 0...255 |
| int | целое | 2 | -32768...32767 |
| unsigned int | беззнаковое целое | 2 | 0...65535 |
| long | длинное целое | 4 | -2147483648... 2147483647 |
| unsigned long | беззнаковое длинное целое | 4 | 0...4294967295 |
| float | вещественное | 4 | ±(3,4\*10-38…3,4\*1038) |
| double | вещественное двойной точности | 8 | ±(1,7\*10-308…1,7\*10308) |
| long double | длинное вещественное двойной точности | 10 | ±(3,4\*10-4932... 3.4\*104932) |

Типы данных *char*, *int* u *long* являются типами для хранения целых чисел. Действительно любой символ связан с целым числом – кодом этого символа по так называемой таблице символов ASCII. Этот символ необходим для идентификации нажатой на клавиатуре клавиши или при выводе на принтер.

Различие между целыми типами данных со знаком и без него (модификатор *unsigned*) состоит в том, как преобразуется старший бит целого числа. Для знакового числа старший бит определяет знак числа - если си равен нулю - число положительное, если равен единице - число отрицательное-

Типы *float*, *double* и *long double* являются типами данных для представления чисел с плавающей запятой.

## 2.2. Переменные

Переменная - это область памяти, имеющая имя и предназначенная для хранения меняющихся данных определенного типа,

Прежде чем использовать переменную в программе, необходимо объявить её. Для этого указывается тип данных, а затем имя переменной. В общем виде объявление переменной выполняется таким образом:

<ТипДанных> <ИмяПеременной>;

Например,

int i; /\* объявление целой переменной i \*/

int j,n=5,k1,k2; /\* объявление 4 целых переменных,

инициализация одной из них \*/

long g=10; /\* объявление переменной g типа длинное целое и инициализация её началь­ным значением 10\*/

float a, b=3.14, с;/\* объявление 3 вещест­венных переменных а, b, с, инициализация начально­го значения b \*/

k1=2\*n+1;

k2=3\*j;

Как видно из примера, в одной строке можно объявить сразу несколько переменных одного типа. Имена (идентификаторы) переменных перечисляются через запятую. Объявление переменных должно заканчиваться точкой с запятой.

Вместе с объявлением переменной, ей можно присвоить начальное значение - этот процесс называется инициализацией. Переменные, которые объявлены, но не инициализированы, содержат случайные значения. Так, в выше приведенном примере после объявления переменных выполнены расчеты значений переменных k1 и k2. После расчета k1 будет иметь определенное значение и равно 11, значение же переменной k2 произвольное, так как в выражении значение переменной j неопределенно.

## 2.3. Имена переменных

Идентификаторы могут содержать буквы верхнего и нижнего регистров латинского алфавита:

A, B, C, ..., Z, a, b, c, ..., z ,

символ подчеркивания:

\_

а также цифры:

0, 1, 2 ..., 9.

Нельзя в имени использовать русские буквы, специальные символы, знаки препинания, пробелы. Обязательным также является следующее условие: имя не должно начинаться с цифры.

Максимально допустимая длина имени переменной различается в зависимости от реализации языка. Язык ТС различает первые 32 символа имени переменной, хотя длина может быть и большей. Еще одной особенностью является то, что имена переменных, написанные строчными и прописными буквами, определяются компилятором ТС как различные, например идентификаторы:

NUM Num num

определяют три различные переменные. При написании программ на языке Си стараются придерживаться следующего правила объявления имен переменных - имя записывается с использованием строчных букв (в отличие от констант, записываемых прописными буквами). Это облегчает чтение и понимание программ. Ниже приведены варианты правильно записанных и неверных именах переменных

|  |  |
| --- | --- |
| Правильные имена | Неправильные имена |
| *first* | *первый* |
| *massiv1* | *1massiv* |
| *hot\_key* | *hot-key* |
| *you\_and\_me* | *you.and.me* |
| *number\_5* | *number№5* |
|  | *printf* |
|  | *if* |

Два последних примера неправильных имен показывают, что нельзя в именах использовать ключевые слова языка ( *if* ) и названия функций ( *printf* ).

Кроме этого, хорошим стилем написания программы считается использование понятных имен переменных. Например, если в программе используется переменная «режим экрана», то можно конечно использовать имя *m* или *r*, но более наглядным будет имя *mode\_screen* или *rejim\_ekran*. К тому же при объявлении переменных желательно использовать в тексте программы комментарии, раскрывающие назначение введенных переменных.

## 2.4. Локальные и глобальные переменные. Область видимости переменной

В языке Си имеет очень важное значение имеет место в программе, в котором происходит объявление переменной. Это определяет область существования переменной (эту область называют областью видимости). Программа не будет компилироваться, если не описать все используемые переменные.

Различают локальные и глобальные переменные. Локальные переменные существуют и в теле функции или оператора управления. Локальные переменные на языке Си (но не С++) объявляются в начале блока – после открывающей фигурной скобки. Только после этого можно использовать их имена в выражениях и операторах. Область видимости локальной переменной – до соответствующей закрывающей фигурной скобки.

Глобальные переменные объявляются вне функций и существуют от точки объявления до конца программы. Соответственно, значение глобальной переменной можно использовать во любом месте программы.

Рассмотрим пример

#include <stdio.h>

int i=1; /\* объявление и инициализация глобальной переменной \*/

float f; /\* объявление глобальной переменной \*/

void fun(void); /\* прототип функции пользователя \*/

void main(void)

{

int k,n; /\* объявление локальных переменных \*/

float g1; /\* объявление локальной переменной \*/

...

}

void fun(void) /\* описание функции пользователя \*/

{

int j=5; /\* объявление локальной переменной \*/

char ch; /\* объявление локальной переменной \*/

...

}

В программе объявлены глобальные и локальные переменные, а также помимо функции main() используется подпрограмма fun(). Глобальные переменные i и f можно использовать в обеих функциях. Значения локальных переменных r, n и g1 можно использовать только в теле функции main(), а переменных j и ch – в теле функции fun().

## 2.5. Константы

В языке Си представляют собой постоянные параметры, константы, которые не изменяются в ходе исполнения программ. Константы могут быть для любого типа данных. Примеры констант приведены в табл. 2.2.

Пример констант различных типов данных

Таблица 2.2.

|  |  |  |
| --- | --- | --- |
| Имя константы | Тип | Примеры |
| символ | char | ’A’ ‘ч’ ‘\n’ |
| целое | int | 10 -9345 32000 |
| длинное целое | long | 100000 5L |
| вещественное двойной точности | double | 3.14 1.23456789 –8.9е-40 |

### 2.5.1. Константы типа символ

Можно задавать в двух формах в виде отдельного символа, который записывается в одиночных кавычках ('А ') или в виде целого числа (55). При этом необходимо помнить, что константы, например, '0' и 0 описывают различные данные.

В языке Си существуют символьные константы, обозначающие специальные управляющие символы, такие как: перевод строки, звуковой сигнал, горизонтальная табуляция. Эти символы нельзя ввести с клавиатуры. Для их использования вначале записывается символ '\' – обратный слеш. Например, символ, означающий перевод строки, описывается так: '\n', горизонтальная табуляция – '\t'. Перечень основных специальных символов приведен в таблице 2.3.

Специальные символы

Таблица 2.3.

|  |  |
| --- | --- |
| Обозначение | Назначение |
| \n | перевод строки |
| \r | возврат каретки |
| \t | горизонтальная табуляция |
| \v | вертикальная табуляция |
| \b | стирание символа, возврат на шаг |
| \f | перевод страницы |
| \” | двойная кавычка |
| \’ | апостроф, одиночная кавычка |
| \\ | обратный слеш |
| \0 | нулевой символ, конец строки |
| \а | звуковой сигнал |
| \? | символ вопроса |

### 2.5.2. Целые константы

Эти константы не имеют десятичной точки и порядка числа. В программировании очень часто используются числа, записанные в других системах счисления - двоичной, восьмеричной и шестнадцатеричной. Для двух последних систем в языке Си есть специальная форма записи:

– для восьмеричных целых констант перед числом записывается ноль, и далее используются цифры 0…7;

– для шестнадцатеричных целых констант перед числом устанавливаются два символа 0х и далее используются цифры 0... 9, а также буквы латинского алфавита A, B, C, D, E, F обозначающие соответственно числа от 10 до 15. Примеры целых констант:

восьмеричные: 01 077 012340

шестнадцатеричные: 0х51 0х6А0 0xF59В

### 2.5.3. Вещественные константы

на языке Си по умолчанию воспринимаются программой как числа типа double. Их можно записать в двух формах:

– в виде числа с десятичной точкой, например, 0.1, 15.189;

– в научном виде (экспоненциальной форме), в котором вместо показателя основания записывается символ Е (или е), например: 1.1Е–10, –5е2 (что эквивалентно математической форме записи чисел 1.1⋅10–10 и –5⋅102).

Различие между числами float и double состоит не только в большем диапазоне последних, но и в более точном значении чисел типа double – верными являются 15...16 значащих цифр.

## 2.6. Макроопределения

Наряду с использованием в тексте программ констант в языке Си существует возможность применения макроопределений (макросов). Макроопределение устанавливает соответствие между именем и определённым значением. Для использования макросов существует несколько предпосылок. Если в программе используется одна и та же константа несколько раз в различных местах программ, то становится желательным использовать соответствующее макроопределение. Другой вариант применения макросов - использование хорошо понимаемых имен.

В общем случае форма записи макроопределения следующая:

#define <ИмяМакроса> <3аменяемоеВыражение>

где <ИмяМакроса> – любое разрешенное имя переменной, при этом желательно использовать заглавные буквы (что облегчает чтение программы). Примеры макросов:

#define NUNBER 1OO

#define LEFT 75

#define RIGHT 77

#define ENTER 13

#define SPACE 0x20

#define PI 3.14159

После определения этих макросов в заголовке, в тексте программы можно использовать имена NUMBER, LEFT, RIGHT и т.д. На этапе создания исполняемого файла (точнее в фазе препроцессора, выполняемой до компиляции) происходит замена имен макросов на соответствующие им значения. Например, если в тексте программы встречаются имена NUMBER, то происходит замена этого имени на значение 100.

## 2.7. Преобразование типов данных

Исходно предполагается, что пользователь правильно использует типы констант и переменных. Что произойдет, если в выражениях присутствуют данные различных типов? В этом случае язык Си выполняет автоматическое приведение типов данных. Например, какое число будет записано в переменной а после инициализации

int а=11.54;

Так как переменная а определяется как переменная целого типа, то дробная часть вещественной константы 11.54 будет отброшена без округления и в ячейке переменной а будет записано целое число 11.

То же самое произойдет, т.е. не будет учитываться дробная часть числа, если исполнить следующие операторы:

float i;

i=1/10;

Это происходит, потому что в начале вычисляется правая часть выражения, а она содержит только целые константы 1 и 10, следовательно, результат тоже будет целое число: 1/10=0, а не 0.1 как ожидалось. Окончательно переменной i будет присвоено значение 0 и это несмотря на то, что она определяется вещественной.

В общем случае при расчете арифметических выражений приведение типов данных в языке Си происходит следующим образом:

1) все переменные типа *char* преобразуются в тип *int*, переменные типа *float* - в тип *double*.

2) для любой пары операндов (т.е. в результате вычисления, например, суммы или произведения двух чисел) происходит следующее преобразование:

– если хотя бы один из операндов *double*, то и второй преобразуется в этот тип;

– если хотя бы один из операндов *long*, то и другой преобразуется в *long*;

– если хотя бы один из операндов *unsigned*, то и другой преобразуется в *unsigned*.

3) конечный результат после операции присваивания приводится к типу переменной в левой части выражения.

Рассмотрим пример вычисления арифметического выражения:

char а=25, b=20;

int i,j,

long k;

double d=l.55;

i=a\*b;

k=10000\*b;

j = b\*d;

printf(“i=%d k=%ld j=%d”,i,k,j);

При вычислении значения переменной i хотя обе переменные имеют тип char и как будто бы результат перемножения должен выйти за максимальные размеры, но по первому правилу обе переменные приводятся к целому типу. В этом случае результат за допустимые пределы не выйдет i=25\*20=500. Для второй переменной k целого типа результат выходит за пределы целого, хотя переменная k объявлена типа длинного целого, поэтому результат перемножения будет неверный. Чтобы исправить эту ошибку необходимо строчку переписать следующим образом:

k=(long)10000\*b;

Для третьей переменной j результатом перемножения будет вещественное число двойной точности, при присвоении этого результата происходит приведение типа (double)→(int) и при этом происходит потеря дробной части полученного произведения. пределы.

## 2.8. Задания для решения

2.8.1. Составьте программу, в которой объявляются две целые переменные a1 и a2, вещественная переменная b и переменные типа символ c и d. Переменная a2 инициализирована значением 10.

2.8.2. Найдите ошибки в приведенной программе

#include <conio.h>

#include <studio.h>

void main() {

int printf=5; /\* объявление целой переменной \*/

long number-one; объявление переменной типа длинное целое

double f=1e,

scanf=3.14;/\* объявление вещественных переменных \*/

char if\_else='a',

red%bull=100;/\* объявление переменных типа символ \*/

unsigned float a=5,

void=1.e-10;;/\* объявление вещественных переменных \*/

getch(); /\* ожидание нажатия любой клавиши \*/

}

2.8.2. Напишите программу, в которой вводится переменная типа символ, она инициализируется значением 125. Прибавьте единицу к переменной и выведите новое значение переменной на экран. Повторите эти операции (увеличения на единицу и вывода на экран) еще 3 раза. Какие значения переменной будут выведены? Объясните результат.

2.8.4. Составьте программу, в которой сообщение "Привет мир" на экран бы выводилось в кавычках.

2.8.5. Запишите в тетради с указанием всех пробелов и созданных новых строк, что будет выведено на экран следующими функциями (символ ␣ означает символ пробела):

printf(“Это\n␣Просто␣␣\nСтрока\n\n”);

printf(“H\n␣E\n␣␣L\n␣␣␣L\n␣␣␣␣O\n”);

printf(“\t\tИванов\n\tПетров\n\Сидоров\n”);

# 3. ФУНКЦИИ ВВОДА-ВЫВОДА

## 3.1. Простейшие функции ввода-вывода

Язык Си содержит множество стандартных функции ввода–вывода информации (с клавиатуры, на экран, из файла и в него и т.д.). В данной главе рассматриваются только самые простые функции – ввода информации с клавиатуры и вывода на экран: printf, scanf, cprintf, cscanf, getch. Эти функции определены в стандартной и консольной библиотеках: <stdio.h> и <conio.h>.

## 3.2. Функция форматированного вывода printf()

### 3.2.1. Назначение и описание функции

Функция позволяет вывести информация на экран: простые сообщения, значения переменных и специальные символы (перевод строки, горизонтальная табуляция и т.д.). Функция определяется в библиотеке <stdio.h>. Синтаксис функции:

printf(<управляющая строка>, <список переменных>);

где <управляющая строка> - строка, которая может содержать:

а) простые сообщения, выводимые на экран без изменения, эти сообщения могут содержать произвольные символы, в том числе и на русском языке;

б) спецификации формата вывода переменных;

в) специальные символы;

<список переменных> - перечень идентификаторов переменных, значения которых необходимо вывести на экран.

Спецификации формата начинаются с символа %, за которым следует код формата. Перечень основных спецификаций приведен в табл. 3.1.

Таблица 3.1.

|  |  |
| --- | --- |
| Формат спецификации | Назначение |
|  | **ЦЕЛЫЕ ЧИСЛА** |
| %d | целое десятичное число |
| %x | целое шестнадцатеричное число |
| %o | целое восьмеричное число |
| %u | целое беззнаковое десятичное число |
| %ld | число типа длинное целое |
|  | **ВЕЩЕСТВЕННЫЕ ЧИСЛА** |
| %f | вещественное число в формате xx.xxxxxx |
| %e | вещественное число в научном формате xx.xxxxx e+xx |
| %g | вещественное число в форматах %f или %e (в какой форме запись будет короче) |
| %lf | вещественное число двойной точности |
|  | **ПРОЧЕЕ** |
| %c | символ |
| %s | строка |
| %p | указатель |

Специальные символы, приведенные в табл. 2.3, можно использовать для создания новой строки, горизонтальной табуляции подачи звукового сигнала и т.д.

Замечание: При использовании функции printf() необходимо быть очень внимательным к соответствию спецификаций форматов в управляющей строке и типов переменных в списке. В противном случае выводимые значения будут существенно отличаться от реальных. Кроме этого, требуется проверять количество вводимых спецификаций и количество переменных в списке – они должны быть равны друг другу.

Примеры:

1) Необходимо вывести на экран сообщение «Значения переменных равны:» и далее текущие значения двух целых переменных a и b, вещественной переменной c. После вывода значения переменных a и b необходимо перевести курсор на новую строку. Эта задача выполняется следующим вызовом функции:

printf(“Значения переменных равны: %d %d\n%f”,a,b,c);

2) Значения двух вещественных переменных x и y, а также символьной переменной symb необходимо вывести на экран в следующем виде (символы … означают, что здесь будет выведено текущее значение переменной):

x=…

y=…

symb=…

т.е. после вывода каждой переменной необходимо выполнить перевод строки. После вывода значений переменных необходимо выдать звуковой сигнал. Эта задача выполняется следующей строкой программы:

printf(“ x=%f\n y=%f\n symb=%c\n\a”,x,y,symb);

### 3.2.2. Точность и ширина поля вывода

В функции printf() существует возможность задания вывода переменных определенного формата, а именно, можно задавать ширину поля переменных, определять точность вывода вещественных переменных, определять тип выравнивания данных и т.д. В этом случае спецификации формата записывается в такой форме:

%[флаг][ширина].[точность]спецификация

где флаг – тип выравнивания (пусто – выравнивания по правому краю, `–` – выравнивания по левому краю, `+` – всегда печатать знак числа);

ширина – общая длина поля вывода, т.е. общее количество символов при выводе, включая знак и десятичную точку;

точность – количество цифр числа после запятой.

Примеры спецификаций формата:

%12.4f – вывод вещественной переменной с выравниванием по правому краю, общее количество символов при выводе –12 позиций, количество знаков после десятичной точки – 4;

%-+10.0f – вывод вещественной переменной с выравниванием по левому краю и обязательным выводом знака числа, общая длина поля вывода – 10 позиций, знаки после десятичной точки не выводятся;

%08d – вывод целой переменной с выравниванием по правому краю, ширина поля – 8 позиций, слева число дополняется нулями;

%16s – вывод строки, ширина поля вывода – 16 позиций.

## 3.3. Функция форматированного ввода scanf()

### 3.3.1. Назначение и описание функции

Функция позволяет ввести информация с клавиатуры. Функция определяется в библиотеке <stdio.h>. Синтаксис:

scanf(<управляющая строка>, <список адресов переменных>);

где <управляющая строка> - строка, которая может содержать только спецификации формата, перечень допустимых значений спецификаций тот же самый, что и для функции printf() (см. табл. 3.1.);

<список адресов переменных> – содержит перечисленные через запятую адреса переменных, вводимых функцией. Адрес переменной указывается символом & и далее идет идентификатор переменной, например, адрес переменной flag обозначается символами &flag. Для массивов и строк указывать символ & не обязательно.

Замечание: При использовании функции scanf() в управляющей строке нежелательно использовать никаких лишних символов (даже пробелов), кроме спецификаций. Нельзя использовать и специальные символы.

Примеры:

1) Ввести с клавиатуры переменные population и age типов длинное целое и вещественное число:

scanf(“%ld%f”, &population, &age);

2) Ввести с клавиатуры целую переменную kod в шестнадцатеричном формате, строку name и вещественную переменную двойной точности precision:

scanf(“%x%s%lf”, &kod, name, &precision);

При вводе нескольких переменных разделителями между ними являются клавиши пробела, табуляции и ENTER.

### 3.3.2. Пример 3.1

Рассмотрим программу, в которой вычисляется значение функции y=sin(x) для двух введенных с клавиатуры значений аргумента. При этом вывод на экран необходимо выполнить в виде следующей таблицы (ширина поля каждой переменной 8 позиций, точность – 4 знака):

|  |  |
| --- | --- |
| x | y |
| xx.xxxx | xx.xxxx |
| xx.xxxx | xx.xxxx |

***Листинг 3.1***

/\* Программа вычисления функции y=sin(x) \*/

#include <stdio.h>

#include <math.h>

void main()

{

float x1,x2,y1,y2;

printf(“введите два значения аргумента:\n”);

scanf(“%f%f”,&x1,&x2);

printf(“\n| x | y |”);

printf(“\n|––––––––|––––––––|\n”);

y1=sin(x1);

y2=sin(x2);

printf(“|%8.4f|%8.4f|\n”,x1,y1);

printf(“|%8.4f|%8.4f|\n”,x2,y2);

}

## 3.4. Функции консольного ввода-вывода cscanf() и cprintf()

Консольная библиотека <conio.h> имеет очень похожие функции ввода-вывода. Отличие состоит в том, что с помощью консольных функций можно задать атрибуты вводимых/выводимых символов: задать положение курсора, изменить цвета фона и символов. Поэтому, например, не имеет смысла использовать некоторые специальные символы при выводе данных на экран (перевод строки, горизонтальная табуляция и т.п.). Синтаксис функций cscanf() и cprintf() идентичен аналогичным функциям библиотеки <stdio.h>.

### 3.4.1. Вспомогательные функции консольной библиотеки

Рассмотрим вспомогательные «полезные» функциями консольной библиотеки, используемые вместе с функциями ввода-вывода:

1) очистка экрана

clrscr();

2) задание нового текущего положения курсора на экране

gotoxy(x,y);

где x, y – координаты нового положения курсора, диапазоны возможных изменения координат следующие x=1…80, y=1…25, координате (1,1) соответствует верхний левый угол экрана;

3) изменение текущего цвета символов (в текстовом режиме):

textcolor(c);

где c – цвет символов (с=0…15 или цвета в виде макроопределения, например, BLACK, BLUE, GREEN, RED,…YELLOW, WHITE);

4) изменение текущего звена фона под символами (в текстовом режиме):

textbackground(c);

где c – цвет фона (с=0…7 или цвета в виде макроопределения, например, BLACK, BLUE, …LIGHTGRAY);

### 3.4.2. Пример 3.2

Создадим программу ввода фамилии в окне. С помощью консольных функций необходимо выполнить следующие операции: очистить экран дисплея, в центре экрана вывести изображение окна (цветная рамка с желтым цветом символов и синим цветом фона) размером 20 столбцов на 3 строки, ввести строку в пределах окна.

***Листинг 3.2***

/\* Рисование окна и ввод фамилии \*/

#include <conio.h>

void main()

{

char name[15];/\*объявление массива символов для ввода строки\*/

clrscr(); /\* очистка экрана \*/

textcolor(YELLOW); /\* задание цвета символов \*/

textbackground(BLUE); /\* задание цвета фона \*/

gotoxy(30,11); /\* задание положения курсора \*/

cprintf(“┌────────────────────┐”);

gotoxy(30,12);

cprintf(“│ Введите фамилию │”);

gotoxy(30,13);

cprintf(“│ │”);

gotoxy(30,14);

cprintf(“└────────────────────┘”);

textcolor(CYAN); /\* задание цвета символов \*/

gotoxy(35,13);

cscanf(“%s”,name); /\* ввод строки \*/

}

Примечание: для ввода программы необходимы символы псевдографики. При их вводе можно использовать коды символов:

218 ┌ 196 ─ 191 ┐

179 │ 179 │

192 └ 196 ─ 217 ┘

## 3.5. Функция ввода одиночного символа getch()

Функции scanf() и cscanf() требуют аккуратного ввода данных с клавиатуры, незначительная ошибка при вводе информации приводит к сбоям в работе программ. В добротных программах используется посимвольный ввод данных с помощью функции getch(), определяемую в той же библиотеке <conio.h>. Эта функция имеет два варианта применения:

а) для остановки программы до нажатия произвольной клавиши, например, перед ее завершением. В этом случае листинг программы может быть таким

{

…

getch(); /\* Ожидание ввода произвольной клавиши \*/

}

б) получение кода ASCII нажатой клавиши. Для получения кода клавиши необходимо объявить целую переменную и присвоить результат исполнения функции getch() этой переменной. При этом синтаксис использования функции может быть следующим:

{

int i;

…

i=getch(); /\* Ожидание нажатия клавиши и присвоение

ее кода символа переменной i \*/

…

}

### 3.5.1. Пример 3.3

Написать программу, принимающую символ с клавиатуры и выдающую его код на экран

***Листинг 3.3***

/\* программа вывода на экран кода нажатой клавиши \*/

#include <conio.h>

void main()

{

int kod; /\* объявление переменной \*/

clrscr(); /\* очистка экрана \*/

cprintf(“Нажмите клавишу: ”); /\* вывод сообщения \*/

kod=getch(); /\* ввод клавиши \*/

cprintf(“\nКод клавиши: %d\n”); /\* вывод кода клавиши \*/

getch(); /\* ожидание нажатия

произвольной клавиши \*/

}

## 3.6. Задание для самостоятельной работы

3.6.1. Записать отдельные строки программы, используя функции форматированного ввода-вывода:

а) вывод на экран двух переменных типа символ, целой переменной в восьмеричном формате и переменной типа длинное целое (идентификаторы выбрать самостоятельно);

б) вывод на экран значения переменных типа строка (массив символов) family, name, gruppa в следующем виде (после символов … должны идти значения указанных переменных):

Фамилия: …

Имя: …

Группа: …

в) вывести на экран две вещественные переменные в заданном формате: 10 позиций общая ширина поля переменной, 3 позиции – точность;

г) ввод двух переменных типа беззнаковое целое first и second;

д) ввод целой переменной x в шестнадцатеричном формате, вещественной переменной y и вещественной переменной двойной точности z;

е) выполнить ввод вещественных переменных power, moment и velocity, используя функции ввода-вывода в следующем формате (после символов … должен идти ввод указанных переменных):

Параметры двигателя:

мощность двигателя – …

момент двигателя – …

скорость двигателя – …

3.6.2. Записать отдельные строки программы, используя только функции консольного ввода-вывода:

а) очистить экран и далее вывести в центре экрана значения двух целых переменных   
a и b, изменив при этом цвет символа и фона;

б) вывести значения четырех вещественных переменных a, b, c и d в следующем виде (ширина поля каждой переменной – 8 позиций, точность – 0 позиций):

a= …

b=…

c=…

d=…

в) выполнить предыдущий пример, но совмещенный с вводом значений этих переменных.

3.6.3. Написать программу: выполняется очистка экрана, вводится вещественное число, на экран выводится корень из введенного числа (функция корня находится в математической библиотеке и вызывается так: y=sqrt(x); ). Вывод корня выполнить в следующем формате – выравнивание по левому краю, общая ширина поля – 10 позиций, точность – 2 символа после десятичной точки.

3.6.4. Составить программу, в которой: выполняется очистка экрана, рисуется на экране изображение окна, внутри окна выполняется ввод трех параметров двигателя (см. задачу 3.6.1,е).

# 4. ОСНОВНЫЕ ОПЕРАЦИИ ЯЗЫКА

## 4.1. Операции языка Си

В языке существует большое количество операций (операторов). С помощью операций языка выполняются вычисления, сравнения, присваивания, а также ряд специфических задач. Наиболее часто используемые приведены в табл. 4.1. При выполнении операторов учитываются следующие особенности их исполнения:

а) каждой операции соответствует определенное количество операндов:

– унарные операции, имеющие только один операнд, например, операция взятия адреса переменной x запишется так: &x, а операция инкрементирования переменной i: i++;

– бинарные операции, имеющие два операнда, например, простейшие арифметические операции сложения, вычитания и т.д.: a+b или 4\*x;

– единственная трехоперандная операция в языке – операция выполнения условия <операнд1> ? <операнд2> : <операнд3>.

б) некоторые операции могут работать только с определенными типами данных, например, побитовые операции применяются только к целым типам данных;

в) в сложных выражениях, содержащих ряд операций, исполнение большинства операций выполняется слева направо, за исключением, унарных операций и операций присваивания. Если необходимо задать строго определенный порядок исполнения выражения можно использовать круглые скобки;

г) в сложных выражениях последовательность исполнения операций определяется их приоритетом (см. крайнюю правую колонку в табл. 4.1, где 1 – наибольший приоритет исполнения операции).

Таблица 4.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Опера-ция | Назначение | Пример | Прио-ритет |
|  | **Первичные операции** |  |  |
| [] | индекс массива | x[2][0]=5; | 1 |
| () | вызов функции, очередность выполнения | y=sin(6.28\*(x+1)); | 1 |
| . | элемент структуры (прямой доступ) | time.tm\_hour=5; | 1 |
| -> | элемент структуры (косвенный доступ) | time->tm\_hour=5; | 1 |
|  | **Унарные операции** |  |  |
| ++ | инкремент (увеличение на единицу) | i++ | 2 |
| –– | декремент (уменьшение на единицу) | i–– | 2 |
| sizeof | размер в байтах | x=sizeof(int); | 3 |
| (тип) | преобразование типа | y=(float)5/2; | 3 |
| - | унарный минус (изменение знака) | y=-x; | 3 |
| \* | косвенная адресация | int x=\*y; | 3 |
| & | взятие адреса | scanf(“%d”,&x); | 3 |
|  | **Арифметические операции** |  |  |
| \* | умножение | 20\*x | 4 |
| / | деление | x/10 | 4 |
| % | деление по модулю | x%10 | 4 |
| + | сложение | x+20 | 5 |
| – | вычитание | 40–x | 5 |
|  | **Битовые операции** |  |  |
| ~ | побитовое НЕ | y=~x; | 3 |
| << | побитовый сдвиг влево | y=x<<1; | 6 |
| >> | побитовый сдвиг вправо | y=x>>1; | 6 |
| & | побитовое И | y=x&15; | 9 |
| ^ | побитовое исключающее ИЛИ | y=x^15; | 10 |
| | | побитовое ИЛИ | y=x|15; | 11 |

Продолжение таблицы 4.1.

|  |  |  |  |
| --- | --- | --- | --- |
| Опера-ция | Назначение | Пример | Прио-ритет |
|  | **Операции отношения** |  |  |
| < | меньше | if (x<6) | 7 |
| > | больше | if (x>6) | 7 |
| <= | меньше или равно | if (x<=6) | 7 |
| >= | больше или равно | if (x>=6) | 7 |
| == | равно | if (x==6) | 8 |
| != | не равно | if (x!=6) | 8 |
|  | **Логические операции** |  |  |
| ! | логическое отрицание (операция НЕ) | if (!x) | 3 |
| && | логическое умножение (операция И) | if (x>0 && x<6) | 12 |
| || | логическое сложение (операция ИЛИ) | if (x<0 || x>6) | 13 |
|  | Условная операция |  |  |
| ?: | при условии | a>b?10;-10 | 14 |
|  | **Операции присваивания** |  |  |
| = | присваивание | x=5 | 15 |
| \*= | присвоение произведения | x\*=3 | 15 |
| /= | присвоение частного | x/=3 | 15 |
| %= | присвоение остатка | x%=3 | 15 |
| += | присвоение суммы | x+=3 | 15 |
| –= | присвоение разности | x–=3 | 15 |
| <<= | присвоение левого сдвига | x<<=3 | 15 |
| >>= | присвоение правого сдвига | x>>=3 | 15 |
| &= | присвоение И | x&=3 | 15 |
| ^= | присвоение исключающего ИЛИ | x^=3 | 15 |
| |= | присвоение ИЛИ | x|=3 | 15 |
|  | **Прочие операции** |  |  |
| , | запятая | a=1, b=2; | 16 |

## 4.2. Арифметические операции

Операции сложения, вычитания, умножения и деления можно использовать ко всем типам данных. Если операнды одного типа, то результат будет того же типа. Например, вычисляя выражение 12/5, так как операнды целые числа, то в результате получим целое число 2 (округление не выполняется). С другой стороны, если рассчитывается выражение 12./5, результатом будет вещественное число 2.4.

Операция деление по модулю используется при необходимости вычисления остатка от деления целых чисел, например, при вычислении выражения 17%3, результатом будет целое число 2.

## 4.3. Операции присваивания

Эти операторы имеют наименьший приоритет при вычислении выражений, т.е. выполняются в последнюю очередь (за исключением оператора запятая). В языке Си знак равенства означает выполнение операции присваивания. Например, выражение

x=10+3\*4;

означает, что вначале вычисляется выражение слева от знака равенства и затем результат присваивается переменной x. Поэтому выражения

x=x+2;

приобретают при программировании строго определенный смысл: вначале вычисляется выражение справа от знака равенства (к старому значению переменной x прибавляется число 2) и затем полученный результат присваивается переменной x.

Оператор присваивания выполняется справа налево. В языке Си возможно использования выражений с несколькими операторами присваивания, например, допустимо использовать следующее выражение

с=b=5\*(a=5\*45);

Оно выполняется в следующем порядке: вычисляется значение 5\*45, результат присваивается переменной ‘а’, затем эта переменная умножается на 5 и результат присваивается переменной ‘b’, и далее присваивается переменной ‘c’.

Помимо традиционной операции присваивания в языке Си существуют дополнительные операции: +=, –=, \*= и т.п. Например, операции слева и справа в приведенной ниже таблице идентичны

|  |  |
| --- | --- |
| a=a+10; | a+=10; |
| a=a/10; | a/=10; |
| a=a&64; | a&=10; |
| a=a>>1; | a>>=1; |

Достоинства этих операции – они выполняются быстрее, чем операции с обычным присваиванием.

### 4.3.1.Пример 4.1

Пример использования арифметических операций и операций присваивания:

***Листинг 4.1***

#include <stdio.h>

void main()

{

int a=25,b=3,с,d;

float x=25.,y=3,z;

c=a/b;

d=a%b;

printf(“Деление целых чисел: %d\nОстаток: %d\n”,c,d);

z=x/y;

printf(“Деление вещественных чисел: %d %d\n”,c,d);

a=a+10;

y=y\*3;

printf(“Традиционная форма операции присваивания: %d %f\n”,a,y);

a+=10;

y\*=3;

printf(“Присваивание со сложение и умножением: %d %f”,a,y);

}

Результат выполнения программы:

Деление целых чисел: 8

остаток: 1

Деление вещественных чисел: 8.333333

Традиционная форма операции присваивания: 35 9.000000

Присваивание со сложение/умножением: 45 27.000000

## 4.4. Операции отношения

Эти операции позволяют сравнивать друг с другом данные, например:

x>0 y<=x z!=27 g==’\n’

Значения переменных после выполнения операции сравнения не изменяется, каждое выражения проверяется – истинно оно или ложно. В языке Си истинному значению выражения с операцией отношения соответствует единичный результат, ложному – нулевой. Например, выражение

y=x>5;

определяет значение переменной ‘у’ равным единице для ‘x’ равным 25 и равным нулю для ‘х’ равным –5.

Согласно табл. 4.1. наибольшим приоритетом обладает операции ‘<’ и ‘>’, наименьшими равенство и неравенство. Поэтому, например, выражение

(x>5 == y<10)

будет выполняться в следующем порядке:

– сравнение y<10;

– сравнение x>5;

– сравнение на равенство.

## 4.5. Логические операции

В языке Си отсутствуют данные типа булева переменная, но существуют логические операции, в которых можно использовать данные типа символ или целое в любой модификации. При этом подразумевается, что нулевое значение выражения соответствует значению «FALSE» (ложь), а любое другое – значению «TRUE» (истина). Эти операции: логическое умножение && (операция И), сложение || (операция ИЛИ) и отрицание ! (операция НЕ). Результат выполнения этих операций для различных значений целых переменных a и b приведен в табл. 4.2.

Таблица 4.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Значение  переменной a | Значение  переменной b | Результат | | | |
| a&&b | a||b | !a | !b |
| ноль | ноль | 0 | 0 | 1 | 1 |
| ноль | отлично от ноля | 0 | 1 | 1 | 0 |
| отлично от ноля | ноль | 0 | 1 | 0 | 1 |
| отлично от ноля | отлично от ноля | 1 | 1 | 0 | 0 |

Следует отметить, что «истинным» значением переменной считается не только ее положительное значение, но и отрицательное. Если выражение при выполнении логической операции является «истиной», то результату присваивается значение единица, в противном случае – нулевое значение.

### 4.5.1. Пример 4.2

***Листинг 4.2***

#include <stdio.h>

void main()

{

int x=5,y=-3;

printf(“x&&y x||y !x !y x>3 y>0\n”);

printf(“%3d %3d %3d %3d %3d %3d”,x&&y,x||y, !x, !y,x>3,y>-10);

}

Результатом исполнения программы будут строки на экране

x&&y x||y !x !y x>3 y>0

1 1 0 0 1 0

Согласно табл. 4.1. наибольшим приоритетом среди логических операций обладает операция НЕ. Поэтому, например, оператор

z=a&&b||!c&&d;

будет выполняться в следующем порядке:

– операция НЕ переменной c;

– операция И операндов a и b;

– операция И операндов (!c) и d;

– операция ИЛИ операндов (a&&b) и (!c&&d).

Логические операции можно использовать и для реализации сложных неравенств. Например, для составления условия 10<x<20 в языке Си необходимо записать логическое выражение (x>10 && x<20), а математическое выражение x<10∪x>20 можно представить в виде (x<10 || x>20).

## 4.6. Битовые операции

Эти операции позволяют выполнять проверку и изменение отдельных битов в данных целого и символьного типов. Их нельзя использовать для вещественных чисел. Действие битовых операций идентично логическим, но они выполняются над каждым битом данных.

Каждая из битовых операций служит определенным целям. Например, операция & (И) полезна для проверки единичных значений отдельных битов числа, а также для установки определенных битов в «0». Операция | (ИЛИ) полезна для установки в «1» битов. Операция ^ (Исключающее ИЛИ) используется для проверки несовпадения битов. Операции сдвига << и >> позволяют сдвинуть все биты числа влево/вправо на заданное количество разрядов. Недостающие значения битов дополняются нулями, информация в исчезающих битах теряется.

Примеры использования битовых операций приведены в табл. 4.3. Все переменные a, b, z этой таблицы типа unsigned char. Язык Си не умеет работать с данными в двоичном формате, но для упрощения понимания работы каждое число представлено в двоичном формате (например, для первой строки a=20210=110010102, b=15510=100110112).

Таблица 4.3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Операция** | **1 операнд** | **2 операнд** | **Результат** | **Комментарий** |
| НЕ | a=11001010 | b=10011011 | z=~a=00110101  z=~b=01100100 | Инвертируются все биты числа |
| И | a=10011010 | b=11110000 | z=a&b=10010000 | На число a накладывается маска (число b) – выделяются значения 4 старших битов числа а, младшие биты обнуляются |
| ИЛИ | a=10011010 | b=00001111 | z=a|b=10011111 | Четыре младших бита числа a устанавливаются в единичное состояние, старшие биты остаются без изменения |
| Исключающее ИЛИ | a=10011010 | b=00001111 | z=a^b=10010101 | Старшие биты числа a остаются без изменения, младшие – инвертируются |
| Сдвиг вправо | a=10011010 | b=00001111 | z=a<<1=00110100  z=b<<3=01111000 | Выполняется сдвиг вправо всех битов числа a на один разряд, числа b – на три разряда. Происходит потеря старших битов, установленных в «1» |
| Сдвиг влево | a=10011010 | b=00001111 | z=a>>2=00100110  z=b>>1=00000111 | Выполняется сдвиг влево всех битов числа a на два разряда, числа b – на один разряд. Происходит потеря младших битов, установленных в «1» |

Следует отметить, что результатом исполнения битовой операции в отличии от логических операций будет число типа символ или целое, а не логические «0» или «1». Так, например, исполнение выражений b=a&&8 и b=a&8 при различных значениях переменной а даст такой результат:

|  |  |  |
| --- | --- | --- |
| Значение переменной а | Результат для b=a&&8 | Результат для b=a&8 |
| 0 | 0 | 0 |
| 8 | 1 | 8 |
| 16 | 0 | 0 |
| 255 | 0 | 8 |

## 4.7. Операции инкрементирования и декрементирования

Во всех языках программирования очень часто используются операции увеличения и уменьшения значения переменных на единицу – операции инкрементирования и декрементирования. В языке Си существуют две специальные формы записи: ++ и ––, например,

x++;

y--;

Операции

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

идентичны друг другу. Различие состоит в том, что последний вариант записи исполняется быстрее.

Каждая из рассмотренных операций имеет две формы записи – префиксную и постфиксную, например, для переменной ‘x’ увеличение ее значения на единицу можно записать двумя способами: x++ или ++x. Разница в выполнении будет в случае использования инкремента в составе более сложных выражений, например, после выполнения операторов

x=10;

y=x++;

значение переменных будут следующие x=11, y=10 (постфиксная форма предполагает вначале использование значения переменной ‘x’, а затем увеличение его на единицу). Но после выполнения операторов

x=10;

y=++x;

значение переменных будут такие x=11, y=11 (префиксная форма предполагает вначале увеличение значения переменной ‘x’ на единицу, а затем его использование).

## 4.8. Операции вычисления размера объекта и преобразования типа данных

Часто требуется учитывать, сколько занимают те или иные переменные в памяти. Для этого используется оператор sizeof. Для этого оператора существует две формы записи: sizeof (<тип данных>) и sizeof (<выражение>). Результатом выполнения оператора будет целое число – размер в байтах занимаемый указанным типом данных или переменной.

### 4.8.1. Пример 4.3

***Листинг 4.3***

#include <stdio.h>

void main()

{

int a,b,c,d;

float x;

double y;

a=sizeof(char);

b=sizeof(unsigned long);

c=sizeof(x);

d=sizeof(y);

printf(“a=%d b=%d c=%d d=%d”,a,b,c,d);

}

Исполнением программы для версии языка ТС-2.0 будет следующий результат:

a=1 b=4 с=4 d=8

В предыдущем разделе уже рассматривалась операция преобразования типа. Для выполнения этой операции используются ключевые слова стандартных типов данных, взятые в круглые скобки. Например, при выполнении операторов

int x=2, y=5;

float z;

z=y/x;

результатом является значение z=2. Для того чтобы получить точное значение отношения целых чисел ‘x’ и ‘y’ необходимо выполнить операцию преобразования типа

z=(float)y/x;

после этого будет z=2.5.