Глава 1. Язык программирования Си

§1.1. Введение в язык Си

Язык программирования Си — универсальный язык программирования, который завоевал особую популярность у программистов, благодаря сочетанию возможностей языков программирования высокого и низкого уровней.

Язык Си широко применяется при современном профессиональном программировании*. Большинство программистов предпочитают использовать язык Си для своих серьезных разработок потому, что их привлекают такие особенности языка, как свобода выражения мыслей, мобильность и чрезвычайная доступность.

Язык Си наряду с тем, что он позволяет освоить хороший стиль программирования, так же как более простые и менее мощные языки высокого уровня (Бейсик, Паскаль), даёт возможность программисту осуществлять непосредственный доступ к ячейкам памяти и регистрам компьютера, требуя при этом знания особенностей функционирования ЭВМ. В этом Си схож с языком низкого уровня — ассемблером. Поэтому язык Си иногда называют ассемблером высокого уровня, хотя на самом деле он представляет собой гораздо более мощное средство решения трудных задач и создания сложных программных систем.

Позиции языка Си++ в современном мире. Современные языки программирования:

- Си++ — язык системного программирования;
- Java — язык программирования для Internet и мобильных систем;
- Visual Basic — язык разработки Windows-приложений;
- Delphi — объектно-ориентированный язык Object Pascal.

Практически все используемые в мире ключевые программные средства, в том числе компиляторы, операционные системы, СУБД, системы телекоммуникаций написаны на Си++. Несколько примеров: а) практически все программные продукты Microsoft (Windows XP, Office XP, Internet Exploer, MS SQL Server и др.), б) ведущие продукты Adobe Systems (Photoshop, Acrobat и др.), в) базовые компиляторы Sun, г) графическая оболочка KDE для Linux, д) многие компоненты Mac OS X и т.д. Не вызывает сомнений подавляющее превосходство Си++ в области встроенных систем и индустрии компьютерных игр (Doom III, StarCraft и др.). На Си++ реализованы ведущие поисковые Web-системы и крупнейшие Web-порталы: Google, Yahoo, Amazon и др. Создатель Си++ Бьерн Страуструп приводит следующие аргументы в пользу языка: «Си++ является наилучшим языком для многих приложений, где требуется системное программирование, имеются определенные ограничения по ресурсам и выдвигаются серьезные требования к производительности. Одним из примеров служит Google, другим — встроенные системы для миниатюрных устройств».

Язык Си был разработан американцем *Деннисом Ритчи* в исследовательском центре Computer Science Research Center of Bell Laboratories корпорации AT&T в 1972 г. Первоначальная реализация Си была выполнена на ЭВМ PDP-11 фирмы DEC для создания операционной системы UNIX. Позже он был перенесен в среду многих операционных систем, обособился и существует независимо от любой из них. Программы, написанные на языке Си, как правило, можно перенести в любую другую операционную систему или на другой компьютер либо с минимальными изменениями, либо вовсе без них.

Диалекты языка Си. Первое описание языка Си дал его автор Деннис Ритчи совместно с Брайном Керниганом в книге «Язык программирования Си». Однако, описание не было строгим и содержало ряд неоднозначных моментов. Разработчики трактовали язык по-разному. Фактически, долгое время стандартом языка служила его реализация в UNIX. Сейчас существуют десятки реализаций языка программирования Си. Они поддерживают разные диалекты языка.

В 1983 г. при Американском Институте Национальных Стандартов (American National Standart Institute — ANSI) был создан комитет по стандартизации языка Си. В 1989 г. был утверждён окончательный вариант стандарта. Однако на сегодняшний день большинство реализаций языка Си не поддерживают стандарт в полном объёме.

§1.2. Структура программы

Программа на языке Си состоит из одной или более подпрограмм, называемых функциями. Каждая функция в языке Си имеет свое имя. В любой программе одна из функций обязательно имеет имя *main*.

Имя функции — это коллективное имя группы объявлений и операторов, заключенных в фигурные скобки. За именем функции в круглых скобках указываются параметры функции.

Пример функции

```
/* Первая программа на Си. */
#include <stdio.h>
main()
{
   printf("\n Здравствуй, язык Си!");
   /* Вывод на экран сообщения.*/
}
```

Результат работы программы

```
Здравствуй, язык Си!
```

В этой программе имя функции *main*. При выполнении программы, созданной на языке Си, операционная система компьютера всегда передаёт управление в программу на функцию с именем *main*. Обычно, хотя это не обязательно, функция *main* стоит первой в тексте программы. Следующие за именем функции круглые скобки играют важную роль. В них указываются параметры (аргументы), которые передаются в функцию при обращении к ней. В данном случае операционная система не передаёт в функцию *main* никаких параметров, поэтому список аргументов в круглых скобках пустой. В фигурные скобки « { } » заключены описания и операторы, которые обеспечивают вывод на экран компьютера сообщения «Здравствуй, язык Си!».

В общем случае программа содержит несколько функций.

Пример программы из нескольких функций

```
#директивы препроцессора

main()
{
    ...
}

function_1(...)
{
    ...
}

function_2(...)
{
    ...
}

...

function_n(...)
{
    ...
}
```

Функция *main* может вызывать для выполнения любую другую функцию. Функции *function_1*, *function_2*, ..., *function_n* могут вызвать любую функцию, кроме функции *main*. Функцию *main* нельзя вызывать изнутри программы, она является управляющей.

§1.3. Объекты языка Си и их типы

Программа, написанная на языке Си, оперирует с объектами. Они могут быть простыми и сложными. К простым объектам относятся переменные и константы, к сложным — массивы, структуры, очереди, списки и т.д. Каждый объект имеет имя и тип. Обращение к объекту программы осуществляется по его имени (*идентификатору*).

Имя объекта — это последовательность не более 32 символов a—z, A—Z, 0—9 и «_» (подчеркивания). Начальный символ имени не должен быть цифрой. Несмотря на то, что допускается имя, имеющее до 32 символов, определяющее значение имеют только первые 8 символов.

Помимо имени, каждый объект имеет тип. Указание типа необходимо для того, чтобы было известно, сколько места в оперативной памяти будет занимать данный объект.

Основные типы и размеры данных:

1) *char* — символьный, 1 байт;
2) *int* — целый, 2 байта;
3) *short* — короткий целый, 2 байта;
4) *long* — длинный целый, 4 байта;
5) *float* — числа с плавающей точкой, 4 байта;
6) *double* — числа с плавающей точкой двойной точности, 8 байт.

Тип char используется для описания символьных объектов. Типы *short*, *long*, *int* предназначены для описания объектов, значения которых выражаются целыми числами. Типы *float* и *double* предназначены для объектов, значения которых выражаются действительными (вещественными) числами.

В программе должно быть дано объявление каждого объекта с указанием его имени и типа. Описание объекта должно предшествовать его использованию в программе.

Пример объявления объектов

```
int n;     /* Переменная n целого типа. */
float x1; /* Переменная x1 типа с плавающей точкой. */
char a;    /* Переменная a символьного типа. */
```

§1.4. Простые объекты

К простым объектам языка Си относятся константы и переменные.

Константа — это ограниченная последовательность символов алфавита языка (лексема), представляющая собой изображение фиксированного (неизменяемого) объекта.

Константы бывают следующие: 1) *числовые*, 2) *символьные* и 3) *строковые*.

Числовые константы делятся на целые и вещественые.

Целые константы

Виды целых констант показаны в табл. 2.1.

Таблица 1.1

Виды целых констант

Десятичные	Последовательность цифр (0 — 9), которая начинаются с цифры отличной от нуля. Пример: 1, –29, 385. Исключение здесь — число ноль 0
Восьмеричные	Последовательность цифр (0 — 7), которая всегда начинается с нуля. Пример: 00, 071, –052, –03
Шестнадцатиричные	Последовательность шестнадцатеричных цифр (0 — 9 и A — F), которой предшествует присутствует 0x. Пример: 0x0, 0x1, –0X2AF, 0X17

В зависимости от значения целой константы компилятор присваивает ей тот или иной тип (*int*, *long*, *unsigned int*).

С помощью суффикса *U* (или *u*) можно представить целую константу в виде беззнакового целого.

Пример

`50000U` — константа типа `unsigned int`

Константе 50000*U* выделяются 2 байта вместо четырех, как было бы при отсутствии суффикса. В этом случае, т.е. для *unsigned int*, знаковый бит используется для представления одного из разрядов кода числа и диапазон значений становится от 0 до 65535.Суффикс *L* (или *l*) позволяет выделить целой константе 4 байта.

Совместное использование в любом порядке суффиксов *U* (или *u*) и *L* (или *l*) позволяет приписать целой константе тип *unsigned long*, и она займет в памяти 32 разряда, причем знаковый разряд будет использоваться для представления разряда кода (а не знака).

Пример

`OLU` — целая константа типа *unsigned long* длиной 4 байта
`2424242424UL` — константа типа *unsigned long*

Вещественные константы

Константа с плавающей точкой (вещественная константа) всегда представляется числом с плавающей точкой двойной точности, т. е. как имеющая тип *double*, и состоит из следующих частей [2]:

- целой части — последовательности цифр;
- десятичной точки;
- дробной части — последовательности цифр;
- символа экспоненты *e* или *E*;
- экспоненты в виде целой константы (может быть со знаком).
- Любая часть (но не обе сразу) из нижеследующих пар может быть опущена:
- целая или дробная часть;
- десятичная точка или символ *e* (*E*) и экспонента в виде целой константы.

Примеры

```
345.
3.14159
2.1E5
.123E3
4037e-5
```

По умолчанию компилятор присваевает вещественному числу тип *double*.

Если программиста не устраивает тип, который компилятор приписывает константе, то тип можно явно указать в записи константы с помощью следующих суффиксов: *F* (или *f*) — *float* для вещественных, *U* (или *u*) — *unsigned* для целых, *L* (или *l*) — *long* для целых и вещественных.

Примеры:

- 3.14159F — константа типа *float*, занимающая 4 байта;
- 3.14L — константа типа *loung double*, занимающая 10 байт.

Символьные константы

Символьная константа — это один символ или обратная косая черта и символ, заключенные в апострофы (одинарные кавычки), например: 'z', ' \n', ' \t' и так далее. Обратная косая черта (слэш) и символ служат для обозначения управляющих символов, не имеющих графического представления, например, '\n' — переход на новую строку, '\t' — табуляция. Все символьные константы имеют тип char и занимают в памяти по 1 байту. Значением символьной константы является числовое значение её внутреннего кода.

Строковые константы

Строковая константа — это последовательность символов, заключенная в кавычки, например: "Это строковая константа". Кавычки не входят в строку, а лишь ограничивают её. Технически, строковая константа представляет собой массив символов и по этому признаку может быть отнесена к разряду сложных объектов языка Си. Однако, строковую константу удобнее рассмотреть вместе с другими константами.

В конце каждой строковой константы компилятор помещает символ '\0', чтобы программе было возможно определить конец строки. Такое представление означает, что размер строковой константы не ограничен каким-либо пределом, но для определения длины строковой константы её нужно полностью просмотреть.

Поскольку строковая константа состоит из символов, то она имеет тип char. Количество ячеек памяти, необходимое для хранения строковой константы на единицу больше количества символов в ней. Следует отчетливо понимать, что символьная константа и строка из одного символа не одно и то же: 'x' не есть "x". Первое — это символ, использованный для числового представления буквы x, а второе — строковая константа, содержащая символ x и '\0'. Если в программе строковые константы записаны одна за другой через разделители, то при выполнении программы они будут «склеены».

Переменные

Переменная — лексема, представляющая собой изображение изменяемого объекта.

С технической точки зрения, переменная — это область памяти, в которую могут помещаться различные числа (двоичные коды). Любая переменная до её использования в программе должна быть описана, т. е. для нее должены быть указаны тип и имя (идентификатор).

Пример

```
тип_переменной  имя_переменной;
```

Предпочтительно использовать именно такой способ описания, чтобы при необходимости можно было модифицировать имя переменной. Кроме того, в этом случае каждую переменную удобно снабдить комментарием, поясняющим ее смысл.

Пример

```
int i;  /* i - счетчик циклов */
```

Общий случай объявления переменных

```
тип_переменных имя_переменной_1,
               имя_переменной_2,
               . . .
               имя_переменной_n;
```

При объявлении переменных им можно задавать начальные значения — производить инициализацию.

Пример

```
тип_переменной  имя_переменной = значение;
```

Примеры

```
int i=0, k, n, m=1;
float x=314.159E-2, y;
char a='a';
```

§1.5. Операции

Над объектами в языке Си могут выполняться различные операции [2]:
1) арифметические;
2) логические;
3) адресные;
4) операции отношения;
5) операции присваивания.

Результат выполнения операции — всегда число.

Операции могут быть *двухместными (бинарными)* или *одноместными (унарными)*. Двухместные операции выполняются над двумя объектами, одноместные — над одним.

Арифметические операции

Основные двухместные операции, расположенные в порядке уменьшения приоритета:
1) умножение — «*»;
2) деление — «/»;
3) сложение — «+»;
4) вычитание и арифметическое отрицание — «-»;
5) целочисленное деление (вычисление остатка от деления) — «%».

Самый высокий приоритет у операции «умножение», самый низкий у операции «целочисленное деление».

Основные одноместные операции:

1) приращение на единицу — «++»;

2) уменьшение на единицу — «−−».

Результат вычисления выражения, содержащего операции «++» или «−−», зависит от того, где расположен знак операции (до объекта или после него). Если операция расположена до переменной, то сначала происходит изменение значения переменной на 1, а потом выполняется какая-то операция; если — после переменной, то сначала выполняется операция, а потом значение переменной изменяется на 1.

Примеры:

- *a**++*b* — если *a*=2 и *b*=3, то результат вычислений равен 8, а *b*=4;
- *a***b*++ — если *a*=2 и *b*=3, то результат вычислений равен 6, а *b*=4.

Логические операции

Логических операций в языке Си три*:*

1) «&&» — логическое «И» (коньюнкция);

2) «||» — логическое «ИЛИ» (дизъюнкция);

3) «!» — логическое «НЕ» (отрицание).

Логические операции могут выполняться над любыми объектами. Результат логической операции: единица, если выражение истинно; ноль, если выражение ложно. Вообще, все значения, отличные от нуля, интерпретируются как истинные. Логические операции имеют низкий приоритет, и поэтому в выражениях с такими операциями скобки используются редко.

Адресные операции

Адресные операции:

1) определение адреса — «&»;

2) обращение по адресу — «*».

Адресные операции являются унарными.

Операции отношения

Операции отношения:

1) равно — « == »;

2) не равно — « != »;

3) меньше — « < »;

4) больше — « > »;

5) меньше или равно — « <= »;

6) больше или равно — « >= ».

Операции используются при организации условий и ветвлений. Все эти операции вырабатывают результат типа *int*. Если отношение между операндами истинно, то значение этого условия — единица, если ложно — ноль.

Операция присваивания

Операция присваивания выполняется следующим образом:
1) вычисляется выражение в правой части;
2) тип результата преобразуется к типу объекта в левой части;
3) результат записывается по адресу, где находится объект.

Пример

```
объект = <выражение>;
```

§1.6. Ввод и вывод информации

Основной задачей программирования является обработка информации, поэтому любой язык программирования должен иметь средства для ввода и вывода данных. В языке Си нет операторов ввода-вывода; ввод и вывод информации осуществляется через функции стандартной библиотеки. Прототипы данных функций находятся в файле stdio.h. Чаще всего вывод осуществляется через функцию *printf*, а ввод — *scanf*.

Функция printf — функция форматированного вывода. Она переводит данные из внутреннего кода в символьное представление и выводит полученные изображения символов (результатов) на экран дисплея. При этом у программиста имеется возможность форматировать данные, т. е. влиять на их представление на экране дисплея. Возможность форматирования условно отмечена в самом имени функции с помощью литеры *f* в конце её названия (print formatted).

Общая форма записи функции printf()

```
printf("строка_форматов",объект_1,объект_2,...,объект_n);
```

Строка форматов состоит из следующих элементов:
1) управляющих символов;
2) текста, который выводится на экран;
3) форматов, предназначенных для вывода значений переменных различных типов.

Объекты могут отсутствовать. Управляющие символы не выводятся на экран, а управляют расположением выводимых символов. Отличительной чертой управляющего символа является наличие слэша перед ним.

Основные управляющие символы:
1) '\n' — новая строка;
2) '\t' — горизонтальная табуляция;
3) '\v' — вертикальная табуляция;
4) '\b' — возврат на символ;
5) '\r' — возврат на начало строки;

6) '\a' — звуковой сигнал.

Форматы нужны для того, чтобы указывать вид, в котором информация будет выведена на экран. Отличительной чертой формата является наличие символа процент «%» перед ним.

Основные форматы:

1) %*d* — целый формат со знаком;
2) %*u* — целый формат без знака;
3) %*f* — вещественный формат (числа с плавающей точкой типа *float*);
4) %*lf* — вещественный формат (числа с плавающей точкой типа *double*);
5) %*e* — вещественный формат в экспоненциальной форме (числа с плавающей точкой типа *float* в экспоненциальной форме);
6) %*c* — символьный формат;
7) %*s* — строковый формат;
8) %*p* — адресный формат.

Пример

```
printf("\n Здравствуй, язык Си!");
```

Результат работы программы

```
Здравствуй, язык Си!
```

Пример

```
a=5;
printf("\n Значение переменной  a=%d.",a);
```

Результат работы программы

```
Значение переменной a=5.
```

Пример

```
x=2.78;
printf("\n Значение переменной  x=%f",x);
```

Результат работы программы

```
Значение переменной x=2.780000
```

При указании формата можно явным образом указать общее количество знакомест и количество знакомест, занимаемых дробной частью.

Пример

```
y=3;
printf("\n Значение переменной y=%10.7f",x);
```

Результат работы программы

```
 Значение переменной y=3.0000000
```

В программе 10 — общее количество позиций под значение переменной; 7 — количество позиций после десятичной точки.

Функция форматированного ввода данных с клавиатуры scanf выполняет чтение кодов, вводимых с клавиатуры, преобразует их во внутренний формат и передаёт программе. При этом программист может повлиять на правила интерпретации входных кодов с помощью спецификаций форматной строки.

Общая форма записи функции *scanf*

```
scanf ("строка_форматов", адрес_объекта_1,
адрес_объекта_2, ..., адрес_объекта_n);
```

Строка форматов аналогична функции *printf*. Адрес объекта генерируется следующим образом: *&имя_объекта*. Строка форматов и список аргументов для функции обязательны.

Пример программы

```
scanf("%d", &m);
/* Ввести целое число и присвоить */
/* его значение переменной m.     */


scanf("%lf", &x1);
/* Ввести значение переменной x1, */
/* имеющей тип double.            */
```

§1.7. Операторы

Условный оператор if

Общая форма записи

```
if(< выражение>)
   <оператор 1>;
```