Курс «Основы программирования»

Федорук Елена Владимировна ст. преподаватель каф РК-6 МГТУ им.Н.Э.Баумана

Лекция №4

Основные типы данных

Переменные – поименованные данные, которые могут изменяться в процессе выполнения программы.

В языке Си все переменные должны быть объявлены явно.

Объявление переменной — это оператор языка Си, который выглядит следующим образом:

тип идентификатор[=значение];

Тип задается соответствующим ключевым словом, например, **int** или **char**.

Тип данных определяет формат представления переменной в компьютере и множество операций, которые могут выполняться над этой переменной.

Идентификатор — в данном случае это имя переменной. В одном операторе объявления могут быть объявлено несколько переменных одного типа путем перечисления идентификаторов, отделенных друг от друга запятыми.

Пример 1

```
int i, y;
double x, y;
```

Переменные при объявлении могут быть инициализированы.

Инициализация переменной — присвоение начального значения.

Для инициализации переменной необходимо при объявлении переменной определить **значение**.

Пример 2

```
int s=56;
```

На рис. 1 приводятся основные типы данных языка Си.

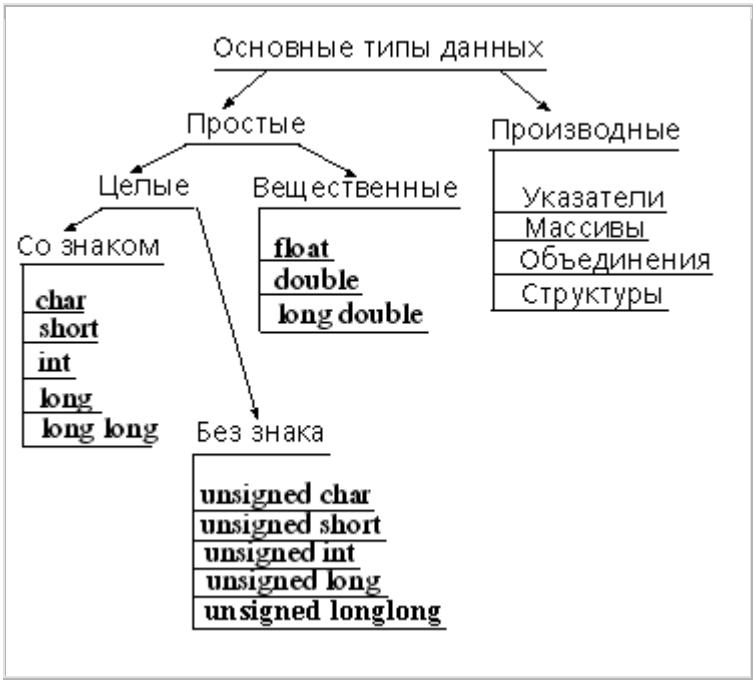

Рис. 1. Основные типы данных языка Си

Размер данных определенного типа, а также *диапазон значений*, которые они могут принимать, зависят от реализации и не зафиксированы в языке Си. Это позволяет компилятору языка Си генерировать программы, выполняемые на конкретном процессоре с максимальной эффективностью. В файлах **<limits.h>** и **<float.h>** содержится информация о фактических размерах и диапазонах значений для различных компиляторов и различных процессоров.

Данные *целого типа* могут быть определены как со знаком, так и без знака. Положительные целые числа хранятся в памяти в прямом коде, отрицательные — в дополнительном коде.

Размер данных типа **char** равен одному байту данного компьютера (как правило 1 байт = 8 бит).

Размер данных типа **short** меньше или равен размеру **int**.

Данные типа **int** обычно имеют размер машинного слова.

Данные типа **long** , соответственно, имеют размер больший или равный размеру **int**.

В общем случае, для компьютера, имеющего размер разрядной сетки, равный n, значение максимального целого числа со знаком (тип **int**) равно $2^{n-1}-1$, а наибольшее беззнаковое целое (тип **unsigned int**) равно $2^{n}-1$.

Вещественные данные хранятся в *формате с плавающей точкой*. Размер данных типа **long double** больше или равен размеру **double**.

Примечание 1

Размер каждого типа данных на конкретном компьютере можно узнать, используя операцию **sizeof()**. Например, **sizeof(int)**.

Размещение в памяти данных целого типа

В прим. 3 показано, как хранятся данные целого типа в памяти гипотетической машины с 8-разрядным байтом и 32-разрядным машинным словом.

Пример 3

```
signed char a = 5, b = -5;
```

Данные типа char занимают один байт – 8 разрядов:
+5 0 0000101 (прямой код)
-5 1 1111011 (дополнительный код)

```
unsigned signed char a = 5;
```

5 00000101

```
signed int a = 5, b = -5;
```

Данные типа int занимают одно машинное слово – 4 байта – 32 разряда:
+5 0 00000000000000000000000000000101 (прямой код)
-5 1 11111111111111111111111111111011 (дополнительный код)

Константы

Константа — определенная в программе и неизменяемая величина.

В программе константа может использоваться как поименованная или как литерал.

Поименованная константа – константа, обращение к которой выполняется по имени.

Она описывается в разделе объявлений переменных и поименованных констант. Для объявления поименованной константы используется ключевое слово **const**.

Пример 1

```
const double pi = 3.14159265;
```

Литерал представляет собой значение константы, записанное непосредственно в тексте программы.

Константы делятся на пять групп:

- *целые константы*;
- *вещественные константы*;
- *константы перечисления*;
- *символьные константы*;
- *строковые константы*.

Компилятор, выделив константу, относит ее к тому или другому типу данных по "внешнему виду" (по форме записи) в исходном тексте и по числовому значению.

Константное выражение — это выражение, оперирующее только с константами. Такие выражения вычисляются во время компиляции, а не во время выполнения программы, и поэтому их можно использовать в любом месте программы, где допустимы константы.

Целые константы

Целые константы в программе на языке Си могут быть представлены в десятичной, восьмеричной или шестнадцатеричной *системах счисления* (СС).

Десятичная константа — целая константа, представленная в десятичной СС.

Восьмеричная константа — целая константа, представленная в восьмеричной СС.

Восьмеричная константа должна начинаться с символа **0** (нуль) и использовать восьмеричные цифры (0, 1, 2, 3, 4, 5, 6, 7).

Шестнадцатеричная константа– целая константа, представленная в шестнадцатеричной СС.

Шестнадцатеричная константа должна начинаться с символов **0x** или **0X** и использовать шестнадцатеричные цифры (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

Обычно целые константы имеют тип **int**.

Длинные целые константы, соответствующие типу **long**, отмечаются символами **L** или **l** на конце.

Пример 1

521 (**52L**)

Константа **521** (**52L**) занимает в памяти столько места, сколько отводится для длинного целого типа **long**. Букву **L** лучше набирать на верхнем регистре, чтобы не перепутать с цифрой 1.

Для обозначения беззнаковых целых констант используется обозначение **U** или **u**.

Пример 2

177536U (**177536u**)

Использование беззнаковых значений особенно полезно при вычислении машинных адресов, такие вычисления должны выполняться с использованием беззнаковой арифметики.

В табл. 1 приведены примеры целых констант.

Таблица 1

Десятичные константы	55	100	255
Восьмеричные константы	067	0144	0377
Шестнадцатеричные константы	0x37	0x67	0Xff
Длинные целые константы	55l	100L	31l
Беззнаковые целые константы	55u	100u	31U

Примечание 1

Не зависимо от того, в каком виде целые константы используются в тексте программы, в компьютере они хранятся в двоичной СС в прямом или дополнительном коде.

Вещественные константы

Вещественные константы - константы, представленные вещественными числами. Их еще называют константами с плавающей точкой.

Вещественные константы могут быть представлены следующим образом:

[знак][целая часть].[дробная часть][E[знак порядка] порядок]

Примечание 1

Часть выражения, записанная в [], может отсутствовать.

По умолчанию вещественные константы имеют двойную точность (тип **double**).

Для указания обычной точности надо использовать **F** или **f** (тип **float**) .

Для указания расширенной точности следует использовать **L** или **l** (тип **long double**).

Вещественные константы в языке Си представляются только в десятичной системе.

В табл. 1 приведены примеры вещественных констант.

Таблица 1

С двойной точностью	5.32	.75	5.	25e-3	.240E100
С обычной точностью	5.32f	.75F	5.f	25e-3F	.240E100F
С расширенной точностью	5.32l	.75L	5.l	25e-3L	.240E100l

Символьные константы

Символьная константа — символ, заключенный в одиночные кавычки (апострофы).

Для обычных печатных символов в апострофах указывается сам символ, (например, **'a' 'B' '+' '5'**). Все символьные константы имеют соответствующее им числовое значение согласно используемой таблице кодирования символов (например, ASCII, КОИ-8) и занимают один байт памяти.

Символьные константы — это целые типа **char**. Они могут участвовать в операциях над числами точно также, как и другие целые, хотя чаще они используются для сравнения с другими символами.

Кроме символьных констант, имеющих отображение, в таблице кодирования присутствуют и специальные управляющие символы, которые записываются с помощью эскейп-последовательностей.

Эскейп-последовательность — комбинация символов, начинающаяся с обратной наклонной черты, которая используется для обозначения трудно представимых или невидимых символов. Такие последовательности изображаются двумя символами, но обозначают один.

Некоторые специальные символы:

- **'\n'** — новая строка

- `'\t'` — горизонтальная табуляция
- `'\b'` - забой
- `'\a'` — звуковой сигнал
- `'\"'` - двойные кавычки
- `'\0'` - нулевой байт
- `'\r'` - возврат каретки
- `'\l'` — прогон страницы
- `'\\'` — обратная наклонная черта
- `'\''` — одиночная кавычка
- `'\?'` — знак вопроса
- `'\"'` — двойная кавычка
- `'\v'` — символ вертикальной табуляции

В качестве символьных констант могут использоваться восьмеричные и шестнадцатеричные значения, состоящие не более чем из трех знаков.

Пример 1

`'\007'` (восьмеричное)

`'\X07'` (шестнадцатеричное)

Восьмеричное и шестнадцатеричное представление кода звукового сигнала в таблице ASCII.

Строковые константы

Строковая константа– это заключенная в двойные кавычки последовательность, состоящая из нуля или более символов. Кавычки не входят в строку, а служат только ее ограничителями. В строку можно также включать эскейп-последовательности. При размещении строковых констант в памяти компилятор Си помещает в конце каждой строки нулевой символ (`'\0'`) в качестве признака конца строки. Строковые константы размещаются в области данных. Фактически строковая константа - это массив символов, последний элемент которого равен нулю.

Разделенные символами пробелов строковые константы автоматически конкатенируются, то есть объединяются или склеиваются.

Пример 1

Две строковые константы `"Смежные строки"` `"объединяются"` будут соединены в одну строковую константу `"Смежные строки объединяются"`. А в конце этой стоки компилятор языка Си поместит признак конца строки `'\0'`.

Значением выражения типа строковая константа является адрес, начиная с которого хранится эта константа.

Константы перечисления

Константы перечисления используются для объявления набора поименованных целых констант.

Для объявления констант перечисления используется ключевое слово **`enum`**.

Формат объявления этих констант:

enum {<Ид1>[=<Целое>][,<Ид2>[=<Целое>]…]} <Список переменных>;

Первое имя (Ид1) имеет значение 0, следующее — 1, и т.д., если для значений констант не было явных спецификаций. Если не все значения специфицированы, то они продолжают прогрессию, начиная от последнего специфицированного значения.

Имена в различных перечислениях должны отличаться друг от друга. Значения внутри одного перечисления могут совпадать.

Пример 1

```
enum {SUN, MON, TUES, FRI=5, SAT} day;
```

Константы присваиваются, начиная с нуля или с указанного значения: SUN=0, MON=1, TUES=2, FRI=5, SAT=6

Атрибут const

Для указания того, что значение переменной не будет изменено в программе, может быть использован *атрибут const*. В этом случае переменная размещается в памяти, доступной только для чтения.

Использование атрибута **const** в прототипе определения функции указывает, что параметр не будет изменяться функцией.

При попытке изменить значение переменной, объявленной с атрибутом **const**, компилятор выдаст сообщение об ошибке.

Пример 1

```
int main()
{
void stringcopy(char [], const char []);
float area_circle(double radius);
. . .
}
void stringcopy(char str1[], const char str2[])
{
. . .
}
float area_circle(double radius)
{
 const double pi = 3.1415;
. . .
}
```

Переименование типа

Оператор typedef — это оператор языка Си, по которому задается новое имя (синоним) существующему типу данных.

Синтаксис оператора **typedef** похож на объявления переменной:

typedef существующий_тип новое_имя

Имена подчиняются тем же правилам, что и прочие идентификаторы.

Хотя объявления **typedef** не являются директивами препроцессора языка Си, они часто используются в файлах заголовков.

Оператор **typedef** может использоваться также для объявления производных типов, таких как, например, матрица 20 на 40 целых значений.

Пример 1

```
typedef char BYTE;
typedef unsigned short USHORT;
typedef int MATRIX[20][40];
typedef int WORD;
int main()
{
BYTE input;
WORD buf [512];
MATRIX prev, current;
. . .
}
```

Примечание 1

Использование заглавных букв в именах типов, определенных оператором **typedef**, — это соглашение, а не правило.