Лекция 4. Типы данных языка Си. Переменные и константы языка Си. Преобразование типов данных

4.1. Типы данных языка Си

Программы в языке Си оперируют с различными данными, которые могут быть простыми и структурированными. Простые данные - это целые и вещественные числа, символы и указатели (адреса объектов в памяти). Целые числа не имеют, а вещественные имеют дробную часть. Структурированные данные - это массивы и структуры.

В языке различают понятия "***тип данных***" и "***модификатор типа***". Тип данных - это, например, целый, а модификатор - со знаком или без знака. Целое со знаком будет иметь как положительные, так и отрицательные значения, а целое без знака - только положительные значения. В языке Си можно выделить пять базовых типов, которые задаются следующими ключевыми словами:

- ***char*** – символьный;
- ***int*** – целый;
- ***float*** – вещественный;
- ***double*** – вещественный двойной точности;
- ***void*** – не имеющий значения.

В таблице 4.1 представлена краткая характеристика типов данных языка Си.

Таблица 4.1.

Краткая характеристика типов данных языка Си

Тип данных	Характеристика
char	Переменная типа ***char*** имеет размер 1 байт, ее значениями являются различные символы из кодовой таблицы, например, '**ф**', '**:**', '***j***' (при записи в программе они заключаются в одинарные кавычки).
int	Размер переменной типа ***int*** в стандарте языка Си не определен. В большинстве систем программирования размер переменной типа ***int*** соответствует размеру целого машинного слова. Например, в компиляторах для 16-разрядных процессоров переменная типа ***int*** имеет размер 2 байта. В этом случае знаковые значения этой переменной могут лежать в диапазоне от -32768 до 32767.
float	Ключевое слово ***float*** позволяет определить переменные вещественного типа, т.е. перменные имеющие дробную часть, например, -5.6, 31.28 и т.п. Вещественные числа могут быть записаны также в форме с плавающей точкой,

	например, -1.09e+4. Переменная типа ***float*** занимает в памяти 32 бита. Она может принимать значения в диапазоне от 3.4e-38 до 3.4e+38.
double	Ключевое слово ***double*** позволяет определить вещественную переменную двойной точности. Она занимает в памяти 64 бита. Переменная типа ***double*** может принимать значения в диапазоне от 1.7e-308 до 1.7e+308.
void	Ключевое слово ***void*** (не имеющий значения) используется для нейтрализации значения объекта, например, для объявления функции, не возвращающей никаких значений.

Объект некоторого типа может быть модифицирован с помощью специальных модификаторов. В языке Си имеются следующие модификаторы: ***unsigned***, ***signed***, ***short***, ***long***.

Модификаторы записываются перед спецификаторами типа**.** Если после модификатора опущен спецификатор, то по умолчанию, спецификатором является ***int***. Таким образом, следующие строки языка Си являются идентичными:

***long a*;**
***long int a*;**

В таблице 4.2 представлены возможные сочетания модификаторов со спецификаторами, а также размер и диапазон значений объекта (для 16-разрядных компиляторов).

Таблица 4.2.

Возможные сочетания модификаторов со спецификаторами в Си

Тип	Размер в байтах (битах)	Интервал изменения
char	*1(8)*	от -128 до 127
unsigned char	*1(8)*	от 0 до 255
signed char	*1(8)*	от -128 до 127
int	*2(16)*	от -32768 до 32767
unsigned int	*2(16)*	от 0 до 65535
signed int	*2(16)*	от -32768 до 32767
short int	*2(16)*	от -32768 до 32767
unsigned short int	*2(16)*	от 0 до 65535
signed short int	*2(16)*	от -32768 до 32767
long int	*4(32)*	от -2147483648 до 2147483647
unsigned long int	*4(32)*	от 0 до 4294967295
signed long int	*4(32)*	от -2147483648 до 2147483647
float	*4(32)*	от 3.4E-38 до 3.4E+38

double	*8(64)*	от 1.7E-308 до 1.7E+308
long double	*10(80)*	от 3.4E-4932 до 3.4E+4932

4.2. Переменные и константы языка Си

Все ***переменные*** до их использования должны быть определены (объявлены). При этом задается тип, а затем идет список из одной или более переменных этого типа, разделенных запятыми.

int a, b, c;
char x, y;

В языке различают понятия объявления переменной и ее определения. Объявление устанавливает свойства объекта: его тип (например, целый), размер (например, 4 байта) и т.д. Определение наряду с этим вызывает выделение памяти (в приведенном примере дано определение переменных). Переменные можно разделять по строкам произвольным образом:

float a;float b;
int a, b;
int c;
char x, y;

Переменные в языке Си могут быть инициализированы при их определении:

***int a* = 25, *h* = 6;**
***char g* = '*Q*', *k* = '*m*';**
***float r* = 1.89;**
***long double n* = *r**123;**

В языке возможны глобальные и локальные объекты. Первые определяются вне функций и, следовательно, доступны для любой из них. Локальные объекты по отношению к функциям являются внутренними. Они начинают существовать, при входе в функцию и уничтожаются после выхода из нее. Ниже показана структура программы на Си и возможные места в программе, где определяются глобальные и локальные объекты:

```
int a;                          //определение глобальной переменной

int function (int b, char c);   //объявление функции

void main (void)
{                               //тело программы
        int d, e;               //определение локальных переменных
        float f;                //определение локальной переменной
}

int function (int b, char c)    /*определение функции и формальных
                                параметров (по существу - локальных
                                переменных) b и c*/
{                               //тело функции
        char g;                 //определение локальной переменной
}
```

Отметим, что выполнение программы всегда начинается с вызова функции ***main*()**, которая содержит тело программы. Тело программы, как и тело любой другой функции, помещается между открывающей и закрывающей фигурными скобками – **{ }**. В языке Си все определения должны следовать перед операторами, составляющими тело функции. Если они сделаны в функции, то соответствующие объекты будут локальными, а если вне функций, то глобальными.

Наряду с переменными в языке существуют следующие виды ***констант***:

- вещественные, например, 123.456, 5.61e-4. Они могут снабжаться суффиксом ***F*** (или ***f***), например, 123.456F, 5.61e-4f;
- целые, например, 125;
- короткие целые, в конце записи которых добавляется буква (суффикс) ***H*** (или ***h***), например, 275h, 344H;
- длинные целые, в конце записи которых добавляется буква (суффикс) ***L*** (или ***l***), например, 361327L;
- беззнаковые, в конце записи которых добавляется буква ***U*** (или ***u***), например, 62125U;
- восьмеричные, в которых перед первой значащей цифрой записывается нуль (0), например, 071;
- шестнадцатеричные, в которых перед первой значащей цифрой записывается пара символов нуль-икс (0x), например, 0x5F;
- символьные - единственный символ, заключенный в одинарные кавычки, например, 'O', '2', '.' и т.п
- строковые - последовательность из нуля символов и более, заключенная в двойные кавычки, например, "Это строковая константа". Кавычки не входят в строку, а лишь ограничивают ее. Строка представляет собой массив из перечисленных элементов, в конце которого помещается байт с символом '\0'. Таким образом, число байтов, необходимых для хранения строки, на единицу превышает число символов между двойными кавычками;
- константное выражение, состоящее из одних констант, которое вычисляется во время трансляции (например, a=60+301);
- типа ***long double***, в конце записи которых добавляется буква ***L*** (или ***l***), например, 1234567.89L.

Объявить константу можно двумя способами:

1. Используется ключевое слово ***const*** **[*type*] <идентификатор>=<значение>**;

const int a*=2;** ⇔ ***const a*=2;** //int*** задается по умолчанию.

2. ***#define*** **<идентификатор>пробел<значение>**

***#define min* 60;**

4.3. Преобразование типов данных

Если в выражении появляются операнды различных типов, то они преобразуются к некоторому общему типу, при этом к каждому арифметическому операнду применяется следующая последовательность правил:

1. Если один из операндов в выражении имеет тип ***long double***, то остальные тоже преобразуются к типу ***long double***.
2. В противном случае, если один из операндов в выражении имеет тип ***double***, то остальные тоже преобразуются к типу ***double***.
3. В противном случае, если один из операндов в выражении имеет тип ***float***, то остальные тоже преобразуются к типу ***float***.
4. В противном случае, если один из операндов в выражении имеет тип ***unsigned long***, то остальные тоже преобразуются к типу ***unsigned long***.
5. В противном случае, если один из операндов в выражении имеет тип ***long***, то остальные тоже преобразуются к типу ***long***.
6. В противном случае, если один из операндов в выражении имеет тип ***unsigned***, то остальные тоже преобразуются. к типу ***unsigned***.
7. В противном случае все операнды преобразуются к типу ***int***. При этом тип ***char*** преобразуется в ***int*** со знаком; тип ***unsigned char*** в ***int***, у которого старший байт всегда нулевой; тип ***signed char*** в ***int***, у которого в знаковый разряд передается знак из ***char***; тип ***short*** в ***int***.

Предположим, что вычислено значение некоторого выражения в правой части оператора присваивания. В левой части оператора присваивания записана некоторая переменная, причем ее тип отличается от типа результата в правой части. Здесь правила преобразования очень простые: значение справа от оператора присваивания преобразуется к типу переменной слева от оператора присваивания. Если размер результата в правой части больше размера операнда в левой части, то старшая часть этого результата будет потеряна.