Уральский федеральный университет

Описание и ввод-вывод данных в языке С

Лекция 4

Иллюстративный материал к лекциям по алгоритмизации и программированию

Автор Саблина Н.Г. 2016 г.

Содержание

Типы данных

Язык С строго типизированный

Все переменные должны быть описаны до их использования, т.е. указан их тип

Тип определяет

- Размер памяти, выделяемой для переменной
- множество значений, которые может принимать переменная,
- множество операций, допустимых над переменной.

Оператор объявления типа данных, имеет вид:

<имя типа> <список переменных>

Типы данных в языке С

В языке С определены 5 базовых типов данных:

- ***char*** - символьный,
- ***int*** - целый,
- ***float*** - с плавающей точкой,
- ***double*** - с плавающей точкой двойной длины,
- ***void*** - пустой, не имеющий значения.

Модификаторы

- ***signed*** - знаковый,
- ***unsigned*** - беззнаковый,
- ***long*** - длинный,
- ***short*** - короткий

Модификаторы применяют

- **signed** и **unsigned** применяются к типам **char** и **int**.
- **short** и **long** - к типу **int**.
- **long** может применяться также к типу **double**.
- **signed** и **unsigned** могут комбинироваться с модификаторами **short** и **long** в применении **int**.

Целые типы

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) или 4(32)	от -32768 (-2^{31})	до 32767 (2^{31}-1)
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
signed long int	4 (32)	от -2147483648	до 2147483647
unsigned long int	4 (32)	от 0	до 4294967295

Действительные типы

Числа с плавающей точкой

Тип	Размер в байтах (битах)	Интервал изменения	
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 1.7E-4932	до 1.7E+4932

Примеры объявлений (описаний) переменных:

- **float** radius;
- **long double** integral;
- **long** LL;

При описании переменные можно инициализировать

- **int** x=0, y, z=1;
- **unsigned char** ch=’q’;

Функции форматного ввода и вывода

Функции **printf ( )** и **scanf ( )**.

Функции могут читать и выводить данные в разном формате, которым можно управлять.

Прототипы функций - в файле STDIO.H

Синтаксис:

printf (“управляющая_строка” , список выводимых аргументов);

scanf (“управляющая_строка” , список вводимых аргументов);

Форматный вывод

Синтаксис:

printf (“управляющая_строка” , список аргументов);

Управляющая строка содержит два типа информации:

- символы, которые непосредственно выводятся на экран,
- команды формата (спецификаторы формата), определяющие, как выводить аргументы.

Команда формата начинается с символа %, за которым следует код формата

Команды формата

- **%f** - десятичное число с плавающей занятой xx.xxxx,
- **%s** - строка символов,
- **%n** - указатель,
- **%c** - символ,
- **%d** - целое десятичное число,
- **%i** - целое десятичное число и т.п.

Управляющие константы при выводе

- \n – переход на новую строку
- \t – горизонтальная табуляция
- \b – возврат курсора на один шаг назад
- \r – возврат каретки
- \a – кратковременная подача звукового сигнала

Примеры форматного вывода

```
int     i=45;

float   x=4.672;

char    c='A';

printf ("\n значение i=%d \t значение x=%f \n удвоенное значение
    i=%d\n значение символа =%c \t код символа=%d", i, x, i*2, c, c);
```

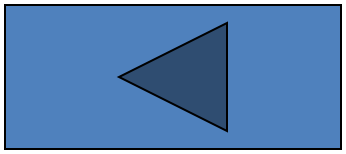
пример

Пример вывода данных

значение i=45 значение x=4.672000

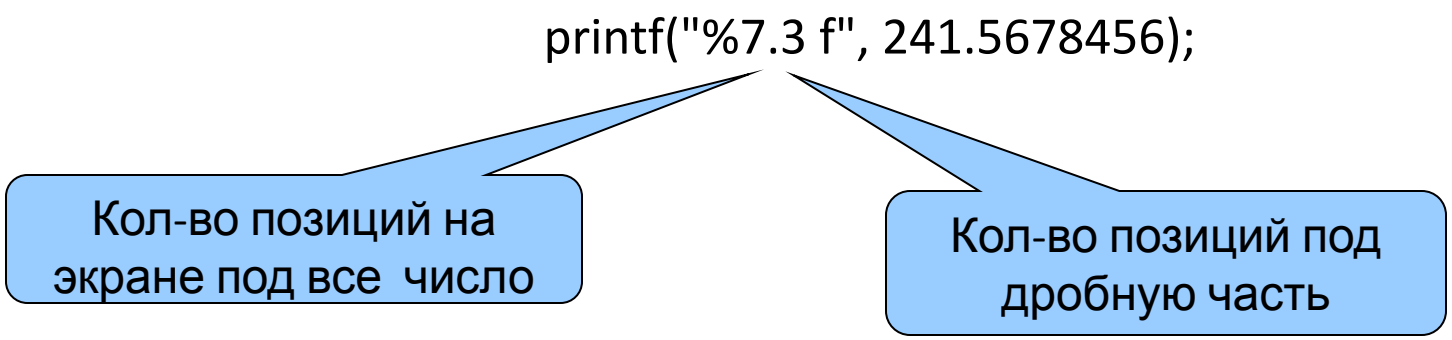
удвоенное значение i=90

значение символа =A код символа=65



Задание ширины полей (1)

- Для действительного числа можно задать точность представления числа



- даст результат 241.568.

Задание ширины полей (2)

Для целого числа можно задать наименьшее поле, отводимое для печати.

Если строка или число больше этого поля, то строка или число печатается полностью, игнорируя ширину поля.

Нуль, поставленный перед целым числом, указывает на необходимость заполнить неиспользованные места поля нулями.

Пример:

```
printf("  целое число %5d, целое с заполнением %05d ", 15, 15);
```

даст результат

```
целое число     15, целое с заполнением 00015
```

Форматный ввод

Синтаксис:

scanf (“управляющая_строка” , список аргументов);

Управляющая строка содержит три вида символов:

- спецификаторы формата
- пробелы
- другие символы .

Спецификатор формата начинается с символа %, за которым следует код формата

Пробел в управляющей строке - пропускает один или более пробелов в потоке ввода. Кроме пробела, может восприниматься символ табуляции или новой строки.

Другой (ненулевой) **символ** указывает на чтение и отбрасывание этого символа

Спецификаторы формата

- **%c** - чтение символа,
- **%d** - чтение десятичного целого,
- **%i** - чтение десятичного целого,
- **%e** - чтение числа типа float,
- **%s** - чтение строки.
- **%p** - чтение указателя,
- и т.п.

Список аргументов

В списке указываются **адреса** аргументов

```
int c; float b;

scanf("%d %f", &c, &d);
```

Знаком & обозначается операция взятия адреса

Разделителями между двумя вводимыми числами являются символы ***пробела, табуляции*** или ***новой строки***.

Список аргументов

Знак * после % и перед кодом формата дает команду прочитать данные указанного типа, но не присваивать это значение.

Пример:

```
scanf("%d%*c%d", &i, &j);
```

При вводе **50+20** присвоит переменной **i** значение **50**, переменной **j** - значение **20**, а символ **+** будет прочитан и проигнорирован.

Ограничение числа вводимых символов

Можно указать наибольшую ширину поля, которая подлежит считыванию.

Пример: `scanf("%5s", str);`

читает из потока ввода первые 5 символов.

При вводе 123456789 строка str будет содержать только 12345, остальные символы будут проигнорированы.

Множество поиска при вводе

Множество поиска определяет набор символов, с которыми будут сравниваться читаемые функцией **scanf()** символы.

Функция **scanf()** читает символы до тех пор, пока они встречаются в множестве поиска.

Как только введенный символ не встретился в множестве поиска, функция **scanf()** переходит к следующему спецификатору формата.

Множество поиска при вводе

Множество поиска - список символов, заключенных в квадратные скобки. Перед открывающей скобкой ставится знак %.

При задании множества поиска можно также использовать символ "дефис" для задания промежутков, а также максимальную ширину поля ввода

Пример 1 использования множества поиска при вводе

```
#include <stdio.h>
/* Форматный ввод с использованием множества поиска */
main (void)
{
char s[10], t[10];
scanf ("%[0123456789]%s", s, t);
printf ("\ns=%s      t=%s", s, t);
}
```

Введем следующий набор символов: 123abc456

На экран программа выдаст

s=123   t=aЬc456

Пример 2 использования множества поиска при вводе

```
#include <stdio.h>
/* Форматный ввод с использованием множества поиска */
main (void)
{
char s[10], t[10];
scanf("%10[A-Z1-5]%s", s, t);
printf ("\ns=%s      t=%s", s, t);
}
```

*Такой формат позволяет вводить в строку **s** заглавные буквы от A до Z, а также цифры от 1 до 5.*

Кроме того, длина строки ограничена 10 символами

Ввод/вывод в C++ (1)

Язык C++ имеет свою библиотеку ввода/вывода.

Она находится в файле **iostream.h**. Этот файл содержит средства управления потоками ввода/вывода.

Ввод с клавиатуры - стандартный входной поток или стандартный ввод - **cin**.

Вывод на экран - стандартный выходной поток или стандартный вывод - **cout.**

Операция вывода **<<** направляет значение в стандартный выходной поток.

cout << index;

Операция ввода **>>** читает значение из стандартного входного потока,

cin >> index;

Ввод/вывод в C++ (2)

Переход на новую строку **cout << endl;** или **cout << '\n';**

например, оператор **cout << "Программа на C++\n";**

В одном операторе вывода можно соединять несколько операций.

Например:

```
cout << "Значение index равно : " << index << endl;
```

Вывод осуществляется по порядку, считая слева направо.

Ввод/вывод в C++ (2)

Операции ввода тоже можно соединять в одном операторе. Например, если в программе встретится следующий оператор :

```
cin >> i1 >> i2;
```

то программа будет ждать ввода с клавиатуры двух величин и первую из них поместит в переменную i1, а вторую - в переменную i2.

Эти две вводимых величины можно разделять пробелом или табуляцией, а можно каждую из них вводить с новой строки - операция ввода сработает правильно.

Итоги

Рассмотренные вопросы:

- Типы данных в С
- Модификаторы
- Ввод/вывод
- функции ввода/вывода
- примеры
- правила

Определение некоторых понятий

- **ANSI** –американский национальный институт стандартизации, организация, авторизированная для создания стандартов в области компьтерии в США
- **ASO** – организация, предназначенная для разработки международных стандартов в компьютерных областях.
- **Аргумент**- выражение, которое задает начальное значение одного из параметров при вызове функции.
- **Стандарт C++** - описание языка программирования, принятого ANSI и ISO для минимизации различий разных реализаций C++ и программ.
- **Тип** – атрибут значения, который определяет его представление и операции, выполняемые над ним, или атрибут функции, который определяет, какие аргументы она ожидает и что она возвращает.

Библиографический список

- Подбельский В.В., Фомин С.С. Курс программирования на языке Си: учебник. М.: ДМК Пресс, 2012. – 384 с.
- Павловская Т.А. C/C++. Программирование на языке высокого уровня: учебник для студентов вузов, обучающихся по направлению "Информатика и вычисл. техника" СПб.: Питер, 2005. - 461 с.
- Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование. Практикум. Практикум. — СПб.: Питер, 2006. — 265 с: ил.
- Березин Б.И. Начальный курс C и C++ / Б.И. Березин, С.Б. Березин. - М.: ДИАЛОГ-МИФИ, 2001. - 288 с
- Каширин И.Ю., Новичков В.С. От С к С++. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2012. – 334 с.

Автор:

Саблина Наталья Григорьевна

Ст. преподаватель

каф. РТС УрФУ