

Информатика

Кочанов Марк

15 сентября 2018 г.

МИФИ

Типы данных

Целочисленные типы данных

Тип	Спецификатор
char (signed char)	%c
unsigned char	%c
short int (signed short int)	%hi
unsigned short int	%hu
int (signed int)	%i или %d
unsigned int	%u
long int (signed long int)	%li
unsigned long int	%lu
long long int (signed long long int)	%lli
unsigned long long int	%llu
float	%f (%e для научной нотации)
double	%lf
long double	%lf

Более подробная информация:

<http://www.cplusplus.com/reference/cstdio/printf/>

<http://www.cplusplus.com/reference/cstdio/scanf/>

Тип	Минимальный диапазон
char (signed char)	8 бит, как правило $[-128, 127]$
unsigned char	8 бит, как правило $[0, 255]$
short int (signed short int)	минимум 16 бит $[-32,767, +32,767]$
unsigned short int	минимум 16 бит $[0, 65535]$
int (signed int)	минимум 16 бит $[-32,767, +32,767]$
unsigned int	минимум 16 бит $[0, 65535]$
long int (signed long int)	минимум 32 бит $[-32,767, +32,767]$
unsigned long int	минимум 32 бит $[0, 65535]$
long long int (signed long long int)	минимум 64 бит $[-32,767, +32,767]$
unsigned long long int	минимум 64 бит $[0, 65535]$
float	%f (%e для научной нотации)
double	%lf
long double	%Lf

Характеристики целочисленных типов (<limits.h>)

- `CHAR_BIT` — размер `char` в битах (минимум 8 бит)
- `SCHAR_MIN`, `SHRT_MIN`, `INT_MIN`, `LONG_MIN`, `LLONG_MIN`(C99) — минимальные возможные значения знаковых целых типов: `signed char`, `signed short`, `signed int`, `signed long`, `signed long long`
- `SCHAR_MAX`, `SHRT_MAX`, `INT_MAX`, `LONG_MAX`, `LLONG_MAX`(C99) — максимальные возможные значения знаковых целых типов: `signed char`, `signed short`, `signed int`, `signed long`, `signed long long`
- `UCHAR_MAX`, `USHRT_MAX`, `UINT_MAX`, `ULONG_MAX`, `ULLONG_MAX`(C99) — максимальные возможные значения беззнаковых целых типов: `unsigned char`, `unsigned short`, `unsigned int`, `unsigned long`, `unsigned long long`
- `CHAR_MIN` — минимальное возможное значение `char`
- `CHAR_MAX` — максимальное возможное значение `char`
- `MB_LEN_MAX` — максимальное число байт в многобайтовых символьных типах.

Ввод/вывод

Примеры вывода

```
1 printf ("Characters: %c %c \n", 'a', 65);
2 printf ("Decimals: %d %ld\n", 1977, 650000L);
3 printf ("Preceding with blanks: %10d \n", 1977);
4 printf ("Preceding with zeros: %010d \n", 1977);
5 printf ("Some different radices: %d %x %o %#x %#o \n",
6         100, 100, 100, 100, 100);
7 printf ("floats: %4.2f %+.0e %E \n",
8         3.1416, 3.1416, 3.1416);
9 printf ("Width trick: %*d \n", 5, 10);
10 printf ("%s \n", "A string");
```

```
Characters: a A
Decimals: 1977 650000
Preceding with blanks:      1977
Preceding with zeros: 0000001977
Some different radices: 100 64 144 0x64 0144
floats: 3.14 +3e+000 3.141600E+000
Width trick:    10
A string
```

Примеры ввода

```
1  char str [80];
2  int i;
3
4  printf ("Enter your family name: ");
5  scanf ("%79s", str);
6  printf ("Enter your age: ");
7  scanf ("%d",&i);
8  printf ("Mr. %s , %d years old.\n",str,i);
9  printf ("Enter a hexadecimal number: ");
10 scanf ("%x",&i);
11 printf ("You have entered %#x (%d).\n",i,i);
```

```
Enter your family name: Soulie
Enter your age: 29
Mr. Soulie , 29 years old.
Enter a hexadecimal number: ff
You have entered 0xff (255).
```


«Новые» арифметические операции

- Compound assignment operator ($+=$, $-=$, $*=$, $/=$, $\%=$)

```
1  int a = 3;  
2  a += 2; // equivalent to a=a+2
```

- Оператор инкремента и декремента

Операция	Синтакс
pre-increment	$++a$
pre-decrement	$--a$
post-increment	$a++$
post-decrement	$a--$

Инкремент и декремент

```
1  int x1 = 10, x2 = 10;
2
3  int y = ++x1 + 1; // y = 12, x1 = 11
4  int z = x2++ + 1; // z = 11, x2 = 11
```

Без учета оптимизации, проводимой компилятором, префиксная форма декремента использует меньшее число инструкций процессора, то есть выполняется быстрее. Делать сложные алгебраические выражения, включающие операции инкремента/декремента, не стоит.

```
1  int i = 10;
2  int x = ++i + i++ + ++i; // i = ?
```

Преобразование типов

Преобразование (приведение) типов (type casting)

- В языках C/C++ неявное (implicit) преобразование типов просходит согласно цепочке: `bool -> char -> short int -> int -> unsigned int -> long -> unsigned -> long long -> float -> double -> long double`.

```
1  int x = 10;
2  double y = 20.5;
3
4  double z1 = x + y;           // z1 = 30.5
5  int z2 = x + y;             // z2 = 30
6
7  int z3 = x + y + y
8  // (10 + 20.5) + 20.5 = 51.0 -> 51
```

- Явное (explicit) приведение типов

```
1  int x = 10, y = 3;
2  float res;
3
4  res = x / y // res = 3
5  res = (float) x / y // res = 3.333333
```

Ошибки при неявном приведении типов

```
1 double a = 10.0;  
2 double b = 1 / 2 * a; // b = ?  
3 double c = 1.0 / 2 * a; // c = ?
```

Константы (литералы)

Символьные константы

```
1 char y1 = 'a'; // y = 97
2 int y2 = 'a'; // y = 97
3 char a = '\\n'; // new string character
4 char b = '\\t'; // tab character
5 char c = '\\\\'; // backslash
```


Целочисленные константы

```
1  int d = 42; // decimal
2  int o = 052; // octal
3  int x = 0x2a; // hex
4  int X = 0X2A; // hex
5
6  // overflow example
7  unsigned int a = 4294967295; // 2^32-1, a = 4294967295
8  unsigned int b = 4294967296; // 2^32, b = 0
9
10 unsigned int x1 = 4000000000;
11 unsigned int x2 = 4000000000;
12 unsigned int x3 = x1 + x2 // x3 = ?
13
14 printf("x3 = %u", x3); // 3705032704
```

Задачи

Задача 1

Реализовать линейный конгруэнтный метод для генерации ряжа псевдослучайных чисел.

Линейный конгруэнтный метод (Д. Г. Лемер, 1949)

$$X_{n+1} = (aX_n + c) \bmod m,$$

где m — модуль (натуральное число, относительно которого вычисляет остаток от деления; $m \geq 2$), a — множитель ($0 \leq a < m$), c — приращение ($0 \leq c < m$), X_0 — начальное значение ($0 \leq X_0 < m$),

Пользователь вводит с клавиатуры пять чисел: количество случайных чисел $nmax$, начальное значение X_0 , множитель a , приращение c и модуль m . На экран через запятую выводится последовательность сгенерированных чисел.

```
8 7 7 7 10
```

```
7, 6, 9, 0, 7, 6, 9, 0
```

Предполагается, что все используемые числа помещаются в переменную типа `int`.

Задача 2

Реализовать алгоритм Евклида нахождения наибольшего общего делителя (НОД, gcd) двух целых чисел

(https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%B0).

Пользователь вводит через пробел два целых числа. На экран выводится НОД.

```
1071 462
21
```

Задача 3

Составить программу, которая будет считывать введённое целое неотрицательное число. Вывести каждую цифру этого числа в новой строке.

```
10819
```

```
1
```

```
0
```

```
8
```

```
1
```

```
9
```