

# Типы данных языка Си

# Концепция типа данных

Язык Си поддерживает *статическую* типизацию.

Тип данных определяет

- внутренне представление данных в памяти;
- множество значений, которые могут принимать величины этого типа;
- операции и функции, которые можно применять к величинам этого типа.

# Классификация типов

## Простые (скалярные) типы

- целый (int);
- вещественный (float и др.);
- символьный (char);
- перечисляемый тип;
- логический тип (с99);
- void;
- указатели.

## Составные (структурированные) типы

- массивы;
- структуры;
- объединения.

# Целый тип

В языке Си существует несколько типов целых чисел. Они различаются

- объемом памяти, отводимым под переменную (диапазоном);
- возможностью присваивания положительных и отрицательных чисел.

Для создания нужного целого типа используется ключевое слово `int` и *модификаторы типа*:

- `signed`
- `unsigned`
- `short`
- `long`

# Целый тип

Только следующие шесть комбинаций «создают» разные типы

- short int
- unsigned short int
- int
- unsigned int
- long int
- unsigned long int

Стандарт C99 добавляет еще два целых типа

- long long int
- unsigned long long int

# Целый тип

Согласно стандарту тип `int` знаковый.

Стандарт определяет

- минимальный диапазон значений (C99 пункт 5.2.4.2, `limits.h`)
- относительные размеры типов (C99 пункт 6.3.1.1)

$$\text{sizeof}(\text{short int}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long int}) \leq \text{sizeof}(\text{long long int})$$

# Целый тип

Тип	Наименьшее значение	Наибольшее значение	Размер в байтах
<b>short int</b>	<b>-32786</b>	<b>32767</b>	<b>2</b>
<b>unsigned short int</b>	<b>0</b>	<b>65535</b>	<b>2</b>
<b>int</b>	<b>-2147483648</b>	<b>2147483647</b>	<b>4</b>
<b>unsigned int</b>	<b>0</b>	<b>4294967295</b>	<b>4</b>
<b>long int</b>	<b>-2147483648</b>	<b>2147483647</b>	<b>4</b>
<b>unsigned long int</b>	<b>0</b>	<b>4294967295</b>	<b>4</b>
<b>long long int</b>	<b>-9223372036854775808</b>	<b>9223372036854775807</b>	<b>8</b>
<b>unsigned long long int</b>	<b>0</b>	<b>18446744073709551617</b>	<b>8</b>

# Целочисленные константы

- Десятичная/восьмеричная/шестнадцатеричная системы счисления.

2016      03740      0x7e0

- Тип целочисленных констант обычно int (компилятор пытается определить наименьший подходящий тип).
- Для указания типа константы используются «суффиксы» L (l), U (u), LL (ll).

2016L      // long int

2016u      // unsigned int

2016ull    // unsigned long long int



# ЧТЕНИЕ И ВЫВОД ЦЕЛЫХ

```
unsigned int u;
```

```
scanf("%u", &u);
```

```
printf("%u", u);
```

```
scanf("%o", &u);
```

```
printf("%o", u);
```

```
scanf("%x", &u);
```

```
printf("%x", u);
```

```
short int s;
```

```
scanf("%hd", &s);
```

```
printf("%hd", s);
```

```
long int l;
```

```
scanf("%ld", &l);
```

```
printf("%ld", l);
```

# Вещественный тип

- Язык Си предоставляет три вещественных типа:
  - float
  - double
  - long double
- Стандарт Си не оговаривает точность указанных типов. Большинство современных компиляторов следует стандарту IEEE Standard 754.
- Константы, определяющие характеристики вещественных типов, могут быть найдены в заголовочном файле float.h.

# Вещественные константы

- Наличие десятичной точки или символа e (или E).

57.0    57.    5.7e1    57E0

- Тип вещественных констант обычно double.
- Для указания типа константы используются «суффиксы» F (f), L (l).

57.0F        // float

57.0l        // long double

# Чтение и вывод вещественных

```
double d;
```

```
scanf("%lf", &d);
```

```
printf("%f", d);    // !!!
```

```
printf("%lf", d);   // c99
```

```
long double ld;
```

```
scanf("%Lf", &ld);
```

```
printf("%Lf", ld);
```

# СИМВОЛЬНЫЙ ТИП

- Для работы с символами предназначен тип `char`. (Под набором символов, как правило, понимается набор символов ASCII.)
- Переменной типа `char` может быть присвоено значение любого ASCII символа.

```
char ch;  
ch = 'a';  
ch = 'A';  
ch = '0';  
ch = ' ';
```

- Под значение типа `char` отводится один байт.

# СИМВОЛЬНЫЙ ТИП

Язык Си интерпретирует символьный тип как «маленькое целое».

```
char ch;  
int i;  
  
i  = 'a';      // 97  
ch = 65;       // A  
ch = ch + 1;   // B  
ch++;          // C  
  
if ('a' <= ch && ch <= 'z')  
    ...  
  
for (ch = 'A'; ch <= 'Z'; ch++)  
    ...
```

Код ASCII символа	ASCII символ
97	a
65	A
66	B
67	C

# СИМВОЛЬНЫЙ ТИП

- Стандарт не определяет «знаковость» этого типа. Если нужно подчеркнуть «знаковость», можно использовать модификаторы `signed` и `unsigned` с типом `char`.

```
signed char sch;  
unsigned char uch;
```

- Для ввода и вывода значений символьного типа используется спецификатор `%c`.

```
char ch;  
  
scanf("%c", &ch);  
printf("%c %d", ch, ch);
```

- Стандартные функции для обработки отдельных символов объявляются в заголовочном файле `ctype.h`.

# Перечисляемый тип

- Тип разработан для переменных, которые принимают небольшое количество значений.
- Значения этого типа *перечисляются* программистом.

```
// Отдельное описание типа
```

```
enum SEASONS {WINTER, SPRING, SUMMER, AUTUMN};
```

```
...
```

```
{
```

```
    SEASONS s;
```

```
// Описание типа совмещено с определением переменной
```

```
enum {WINTER, SPRING, SUMMER, AUTUMN} s;
```



# Перечисляемый тип

Си трактует значения перечислимого типа как целые числа.

```
enum SEASONS {WINTER = 1, SPRING = 2, SUMMER = 3, AUTUMN = 4};
```

```
int i;
```

```
enum {WINTER, SPRING, SUMMER, AUTUMN} season;
```

```
i = WINTER;           // 0
```

```
season = 0;            // 0 (WINTER)
```

```
season++;              // 1 (SPRING)
```

```
i = season + 2;        // 3
```

```
season = 4;            // 4 (?)
```

# Логический тип (C99)

- Стандарт C99 добавил логический тип `_Bool`.

```
_Bool flag;
```

- Переменные типа `_Bool` могут принимать только значения 0 и 1.

```
flag = 5;  
printf("%d", flag);           // 1
```

- Стандарт c99 предоставляет заголовочный файл `stdbool.h`, который облегчает использование «нового» логического типа.

```
#include <stdbool.h>  
...  
bool flag;  
...  
flag = true;
```

# void

- В заголовке функции `void` может указывать на то, что
  - функция не возвращает значение;
  - функция не имеет параметров.
- `void*` - указатель, способный представлять адреса любых объектов.

# Оператор typedef

Позволяет определять имена новых типов.

**typedef тип имя;**

“+” улучшает читаемость.

“+” облегчает внесение изменений.

```
unsigned int n_pens, n_copybooks;
```

```
typedef unsigned int quantity_t;
```

```
quantity_t n_pens, n_copybooks;
```

# Операция sizeof

Операция sizeof возвращает размер переменной или типа в байтах .

**sizeof(выражение);**

```
char ch;  
int i;
```

```
printf("%d\n", sizeof(ch));      // 1  
printf("%d\n", sizeof i);       // 4  
printf("%d\n", sizeof(double)); // 8
```

Операция	Название	Нотация	Класс	Приоритет	Ассоциат.
<b>sizeof</b>	Размер	<b>sizeof X</b>	Префиксная	15	Справа налево