# Программирование и алгоритмизация

Санкт-Петербургский государственный политехнический университет

01 октября 2014

# Типы данных

- Для программиста: целые, плавающие, символы, строки, пользовательские типы.
- Для процессора: совокупность 0 и 1, разному объекту соответствует разное количество битов.

# Типы данных

Для программиста подразделяются на:

- Константы — программист точно знает значение при написании программы. Подразделяются на:
  - Целые: Знаковые, Беззнаковые. (Арифметические)
  - С плавающей точкой. (Арифметические)
  - Символы.
  - Строки символов.
  - Логические.
- Переменные — могут изменять данные в процессе выполнения программы. Подразделяются на:
  - Целые: Знаковые, Беззнаковые. (Арифметические)
  - С плавающей точкой. (Арифметические)
  - Символы.
  - Строки символов.
  - Логические.
  - Адреса.

# Типы данных

Тип константы распознается компилятором по ее записи в тексте программы, тип переменной программист должен указать явно в тексте программы.
C/C++ является языком со строгой типизацией.

# Константы (Литералы)

Литералом называется текстовое представление значения константы. Компилятор переводит литералы в двоичный код с которым работает процессор. Примеры целых литералов:

- десятичная: 8, 10, 256.
- шестнадцатеричная: 0x8, 0xa(0xA), 0X100.
- восьмеричная: 010, 012, 0400.

# Целые константы

- если значение константы помещается в диапазон представления int (2 или 4 байта в зависимости от архитектуры);
- если значение выходит из диапазона, то помещается в переменную long int (4байта);
- если значение находится за пределами long, то столько байт сколько помещается в long long (8 байт);
- если выходит за пределы long long, то отсекается до 8 байт.

# Целые константы

Размер констант может изменяться программистом благодаря указанию суффиксов:

- L или l. 10L, 10l.
- LL или ll. 10LL, 10l.
- 0xffffffffLL + 1LL

Знаковость констант:

- если константа помещается в диапазон знакового, то константа записывается как знаковое число. Например: -1
- иначе константа трактуется как беззнаковая. Поведение можно поменять, если задать суффикс u (U). Например: 255U.

Суффиксы можно комбинировать, например, 123456789123ULL.

# Примеры

- (-1 « n)
- (-1 » n)
- (-1U » n)

# С плавающей точкой

Примеры записи литералов с плавающей точкой:

- 1.25
- 0.25
- .25
- 1.
- 1.25E10 //мантиссаEпорядок
- 1.25e10

По умолчанию все литералы с плавающей точкой относятся к типу double. Возможно добавить суффикс f, который будет означать тип float: 0.25f

# Символьные литералы

Каждому символу, используемому в компьютерных программах соответствует уникальное значение (двоичный код).
Количиство байтов отводимых под символ может различаться в зависимости от способа кодировки.

- ASCII (American Standart Code for Information Interchange) — 7 битов.
- SBCS (Single Byte Character Set) — 8 битов (1байт). 'Q' — 0x51, 'Ф' — 0xd4, '\n' — 0x0A, '\a' — 0x07, '\\' — 0x5c.
- UNICODE — 2 байта. L'Ф' — 0x0424.

# Строковые литералы

- Строковый литерал — это массив, в котором компилятор хранит коды указанных программистом символов.
- При генерации такого массива компилятор автоматически добавляет завершающий нулевой байт, такое значение считается в Си признаком конца строки.
- Строка может быть пустой.
- Строковые литералы могут конкатенироваться: “12” “34”, эквивалентно “1234”
- Строка с префиксом L, считается строкой UNICODE: L“Вася”.
- Длинные строки можно переносить указывая символ \.

# Перечисление

enum — перечисляемый тип, который записывается как:

```
enum [имя_пользовательского_типа] {
    список_именнованых_констант};
```

Тип эквивалентен знаковому целому signed int.

# enum Цвета

```
int n;
cout << "Введите цвет";
cin >> n;
if(n == 0) //красный
else if(n == 1) //синий
```

```
enum {RED, GREEN, BLUE};
int n;
cout << "Введите цвет";
cin >> n;
if(n == RED) {} //красный
else if(n == BLUE) {} //синий
```

# enum Цвета

```
enum Colors {RED, GREEN, BLUE, YELLOW = 100}; //RED - 0, GREEN - 1, BLUE - 2, YELLOW - 100
Colors c = RED;
c = 1; //ОШИБКА!!!
int x = c; //OK
if(c == RED)
{
c = RED | GREEN;
}
```

# Ключевое слово typedef

Позволяет вводить синоним типа. Не создает новых типов, а всеголишь дает им дополнительное имя.
typedef тип синоним_типа;

```
typedef unsigned char byte;
typedef byte Newbyte; //???

unsigned char c1 = 0;
byte c2 = 0;
```