Лекция III

8 октября 2016

Работа с текстом. Ещё раз о символах

```
1  char sym = '5';
2  std::cout << "\n Значение sym: " << sym;
3
4  std::cout << "\n Код sym: " << int(sym) << "↩
       \n";
5
6  bool is_less = '2' < sym;
7  // Переменная is_less здесь равна true
8
9  // Печать всех букв английского алфавита
10 sym = 'a'
11 while (sym <= 'z') {
12   std::cout << sym;
13   ++sym;
14 }
```

Кодировки.

ASCII

- 128 символов
- коды: от нуля до 127
- каждый символ занимает один байт
- Цифры, буквы английского алфавита идут последовательно

UTF8

- символ может состоять от 1 до 4 байт (следовательно, от 8 до 32 бит)
- вводится понятие "code point равное 1 байту (8 битам)
- Включает в себя все возможные региональные алфавиты

Работа с текстом. С-строки

Строка в С - это одномерный массив значений типа **char**, последним символом которого обязательно должен быть равен нулевому символу (он же - символ конца строки): \0.

```
1 char phrase1[] = "Первая строка!";
2 std::cout << phrase1;
3
4 char phrase2[] = { 'A', 'B', 'C', '\0' };
5 std::cout << "\nВторой символ в phrase2: " ↩
     << phrase2[1];
6
7 char test_str[15];
```

Работа с текстом. С-строки. Ввод с консоли

```
1 char prase[20]
2
3 // Небезопасно! Следует всегда избегать
4 std::cin >> phrase;
```

Работа с текстом. С-строки. Ввод с консоли

```
std::cin.getline(str, count)
std::cin.getline(str, count, delimiter = '\n')
```

1. **str** - массив типа **char**
2. **count** - положительное целое число, означающее максимальное число символов, записываемых в str **с учётом символа конца строки** $\backslash 0$.
3. **delimiter** - символ типа **char**, на котором прекращается считывание знаков в строку str.

Работа с текстом. C-строки. Ввод с консоли

```
1  const size_t SZ = 20;
2  ...
3  char phrase[SZ];
4
5  // Вот другое дело
6  std::cin.getline(phrase, SZ);
7
8  char word[SZ];
9  // Считываем все символы до первого пробела
10 std::cin.getline(word, SZ, ' ');
```

Библиотека языка С для работы со строками:

```
1 #include <cstring>
```

```
std::strlen(str)
```

```
1 #include <cstring>
2 ...
3
4 char text[] = "A string";
5 // На экране покажется число 8
6 std::cout << "\nДлина строки: " << text;
7
8 char text2[] = "Строка на русском";
9 // А здесь – 32
10 std::cout << "\nДлина строки: " << text2;
```

Копирование строки

`std::strcpy(destination, source)`

```
1 #include <cstring>
2 ...
3
4 char source[] = "A string", dest[40];
5 std::strcpy(dest, source);
6
7 std::cout << "\n" << dest;
```

Копирование строки

`std::strncpy(destination, source, num)`

num - целое безнаковое число

```
1 #include <cstring>
2 ...
3
4 char source[] = "Просто длинная строка";
5 char dest[15];
6
7 std::strncpy(dest, source, 15);
8
9 std::cout << "\n" << dest;
```

Сравнение строк

```
int std::strcmp(first, second)
```

Функция возращает:

1. значение $<$ **0** - первый несовпадающий символ в строке **first** меньше, чем в строке **second**;
2. значение $==$ **0** - все символы в обоих строках совпадают;
3. значение $>$ **0** - первый несовпадающий символ в строке **first** больше, чем в строке **second**;

Сравнение строк

```
1  #include <cstring>
2  ...
3
4  char str1[] = "First",
5       str2[] = "Second",
6       str3[] = "First";
7
8  std::cout << "Сравнение str1 и str2: ";
9  std::cout << std::strcmp(str1, str2);
10
11 std::cout << "Сравнение str1 и str3: ";
12 std::cout << std::strcmp(str1, str3);
```

Работа с текстом. C++ строки

В стандартной библиотеке языка C++ реализован специальный класс для упрощения работы со строками. Подключается он так:

```
1 #include <string>
```

Какие преимущества даёт по сравнению с прямым использованием массивов типа **char**?

- Автоматическое выделение место под строку.
- Использование привычных операторов сравнения: $>$, $<$, $==$, $!=$, ...
- Нет ограничений при передаче в функции (в отличии от массивов).

Работа с текстом. C++ строки

Определение переменной класса **string**.

```
1 #include <string>
2 ...
3
4 std::string s1 = "Строка 1", s2;
5 s2 = "Строка 2";
6
7 std::cout << s1 << "\n" << s2 << "\n";
8
9 std::string s3 = "English-based string";
10 std::cout << s3[0] << s3[2] << s3[4] << "\n"↩
     ;
11
12 std::string s4 = s1 + s2;
```

Работа с текстом. C++ строки

Ввод строки.

```
std::getline(std::cin, str_to_fill,
             delimiter = '\n')
```

```
1 #include <string>
2 ...
3
4 std::string s1, word, s2;
5
6 // Вполне безопасно
7 std::cin >> word;
8
9 getline(std::cin, s1);
10 getline(std::cin, s2, '*');
```

Сравнение строк.

```
1 #include <string>
2 ...
3
4 std::string s1 = "France",
5              s2 = "Russia";
6
7 if ( s1 < s2 ) {
8   std::cout << "Кто бы сомневался\n";
9 }
10
11 bool is_equals = s1 == s2;
12 // is_equals равен false
```

Работа с текстом. C++ строки

Получение длины (числа байт) строки.

```
size_t str.size()
size_t str.length()
```

```
1 #include <string>
2 ...
3
4 std::string s1 = "France";
5
6 std::cout << "Длина s1: " << s1.size();
```

Работа с текстом. C++ строки

Удаление всего содержимого строки из переменной

```
void str.clear()
```

```cpp
#include <string>
...

std::string s1 = "France";
std::cout << "Длина s1: " << s1.size();

s1.clear();
std::cout << "Длина s1: " << s1.size();
```

Добавление текста к строке

```
1 #include <string>
2 ...
3
4 std::string s1 = "Быть";
5
6 s1 += " или не быть";
7 s1 += '?';
8 s1 += " Вот в чём вопрос!";
9
10 std::cout << s1;
```

Работа с текстом. C++ строки

Вставка на указанную позицию

`str.insert(pos, another_str)`

Вставляет строку **another_str** внутрь **str** сразу перед позицией, заданной **pos**.

```
1 #include <string>
2 ...
3
4 std::string s1 = "Что дела?";
5 s1.insert(7, "за ");
6
7 // Напечатает "Что за дела?"
8 std::cout << s1;
```

Преобразование в C-строку

`str.c_str()`

```
1 #include <string>
2 #include <cstring>
3 ...
4
5 std::string s1 = "Странное сообщение";
6 char c_str[] = "и не говори";
7
8 std::cout << strcmp(c_str, s1.c_str());
```

Работа с текстом. C++ строки

Выделение подстроки

```
string str.substr(start, len = npos)
```

start - переменная типа **size_t**, указывающая позицию первого символа подстроки. **len** - количество символов для извлечения. По умолчанию **len** равна константе **std::string::npos** - это специальное безнаковое целое число, обозначающее конец строки.

```
1 std::string s1 = "Phase transitions are ↩
    great part of physics";
2 std::string s2 = s1.substr(22, 5);
3 std::cout << s2 << "\n";
4
5 std::string s3 = s1.substr(28);
6 std::cout << s3 << "\n";
```

Работа с текстом. C++ строки

Поиск в строке

```
size_t str.find(another_str, pos = 0)
size_t str.rfind(another_str, rpos = npos)
```

Возращает позицию первого символа строки **another_str** в строке **str**.
Поиск начинается с позиции, определяемой вторым аргументом.
rfind - поиск с конца строки.

Работа с текстом. C++ строки

Поиск в строке

```
1  std::string s1 = "Сопротивление обратно ↩
       пропорционально силе тока";
2  size_t found_pos = s1.find("обр");
3
4  if ( found_pos != std::string::npos ) {
5    std::cout << "Позиция \"обр\": " << ↩
         found_pos;
6  }
7
8  found_pos = s1.find("ТЧК", 6);
9  if ( found_pos == std::string::npos ) {
10   std::cout << "\n\"ТЧК\" в исходной строке ↩
         не обнаружена";
11 }
```