Стандартная библиотека

Никитин Михаил Евгеньевич, 2015

Введение

- Ни одна программа приличных размеров не пишется с использованием только «голых» конструкций языка.
- Сначала разрабатываются или выбираются библиотеки поддержки.
- Средства стандартной библиотеки являются частью любой полной реализации C++.
- Стандартная библиотека определена в пространстве имен ***std.***

Ввод/вывод

- Заголовочный файл <***iostream***>.
- Возможность управлять вводом-выводом в C++, обеспечивают форматирующие функции-члены, флаги и манипуляторы.
- Флаги, функции и манипуляторы выполняют одну и туже задачу — задают определённый формат ввода/вывода информации в потоках.

Ввод/вывод

- Ввод/вывод на экран/с экрана в C++ осуществляется с помощью операторов cin и cout соответственно, а значит манипуляторы форматирования используются совместно с данными операторами ввода/вывода.
- Различие между функциями флагами и манипуляторами форматирования состоит в способе их применения. Теперь рассмотрим способы применения объектов форматирования.

Ввод/вывод

- Доступ к функциям осуществляется через операцию **точка**, а в круглых скобочках передаётся аргумент.
- Ещё один способ форматирования — флаги.
- Флаги форматирования позволяют включить или выключить один из параметров ввода/вывода.
- Чтобы установить флаг ввода/вывода, необходимо вызвать функцию setf(), если необходимо отключить флаг вывода, то используется функция unsetf().

Флаги

- Флаги вывода объявлены в классе ios
- Установка флага

```
cout.setf(ios::/*name_flag*/);
```

- Снятие флага

```
cout.unsetf(ios::/*name_flag*/);
```

Флаги

- Если при вводе/выводе необходимо установить(снять) несколько флагов, то можно воспользоваться поразрядной логической операцией ИЛИ |

```
cout.setf(ios::/*name_flag1*/ |
ios::/*name_flag2*/ | ios::/*name_flag_n*/);

cout.unsetf(ios::/*name_flag1*/ |
ios::/*name_flag2*/ | ios::/*name_flag_n*/);
```

Флаги форматирования в C++

Флаг	Назначение	Пример	Результат
boolalpha	Вывод логических величин в текстовом виде (true, false)	**cout.setf(ios::boolalpha); bool log_false = 0, log_true = 1; cout << log_false << endl << log_true << endl;**	**false true**
oct	Ввод/вывод величин в восьмеричной системе счисления (сначала снимаем флаг dec, затем устанавливаем флаг oct)	**cout.unsetf(ios::dec); cout.setf(ios::oct); int value; cin >> value; cout << value << endl;**	ввод:$\mathbf{99_{10}}$ вывод:$\mathbf{143_{8}}$
dec	Ввод/вывод величин в десятичной системе счисления (флаг установлен по умолчанию)	**cout.setf(ios::dec); int value = 148; cout << value << endl;**	**148**
hex	Ввод/вывод величин в шестнадцатеричной системе счисления (сначала снимаем флаг dec, затем устанавливаем флаг hex)	**cout.unsetf(ios::dec); cout.setf(ios::hex); int value; cin >> value; cout << value << endl;**	ввод:$\mathbf{99_{10}}$ вывод:$\mathbf{63_{16}}$

Флаги форматирования в C++

Флаг	Назначение	Пример	Результат
showbase	Выводить индикатор основания системы счисления	**cout.unsetf(ios::dec); cout.setf(ios::oct | ios::showbase); int value; cin >> value; cout << value << endl;**	ввод:**99** вывод:**0143**
uppercase	В шестнадцатеричной системе счисления использовать буквы верхнего регистра(по умолчанию установлены буквы нижнего регистра)	**cout.unsetf(ios::dec); cout.setf(ios::hex | ios::uppercase); int value; cin >> value; cout << value << endl;**	ввод:**255** вывод:**FF**
showpos	Вывод знака плюс + для положительных чисел	**cout.setf(ios::showpos); int value = 15; cout << value << endl;**	**+15**
scientific	Вывод чисел с плавающей точкой в экспоненциальной форме	**cout.setf(ios::scientific); double value = 1024.165; cout << value << endl;**	**1.024165e+003**

Флаги форматирования в C++

Флаг	Назначение	Пример	Результат
fixed	Вывод чисел с плавающей точкой в фиксированной форме(по умолчанию)	**double value = 1024.165; cout << value << endl;**	**1024.165**
right	Выравнивание по правой границе (по умолчанию). Сначала необходимо установить ширину поля(ширина поля должна быть заведомо большей чем, длина выводимой строки).	**cout.width(40); cout << “cppstudio.com” << endl;**	**__cppstudio.com**
left	Выравнивание по левой границе. Сначала необходимо установить ширину поля(ширина поля должна быть заведомо большей чем, длина выводимой строки).	**cout.setf(ios::left); cout.width(40); cout << “cppstudio.com” << endl;**	**cppstudio.com__**

Манипуляторы в C++

- Ещё один способ форматирования — форматирование с помощью манипуляторов.
- Манипулятор — объект особого типа, который управляет потоками ввода/вывода, для форматирования передаваемой в потоки информации.
- Отчасти манипуляторы дополняют функционал, для форматирования ввода/вывода.

Манипуляторы в C++

Манипулятор	Назначение	Пример	Результат
endl	Переход на новую строку при выводе	**cout << «website:» << endl << «cppstudio.com»;**	**website: cppstudio.com**
boolalpha	Вывод логических величин в текстовом виде (true, false)	**bool log_true = 1; cout << boolalpha << log_true << endl;**	**true**
noboolalpha	Вывод логических величин в числовом виде (true, false)	**bool log_true = true; cout << noboolalpha << log_true << endl;**	**1**
oct	Вывод величин в восьмеричной системе счисления	**int value = 64; cout << oct << value << endl;**	**100**

Манипуляторы в C++

Манипулятор	Назначение	Пример	Результат
dec	Вывод величин в десятичной системе счисления (по умолчанию)	**int value = 64; cout << dec << value << endl;**	**64**
hex	Вывод величин в шестнадцатеричной системе счисления	**int value = 64; cout << hex << value << endl;**	**40**
showbase	Выводить индикатор основания системы счисления	**int value = 64; cout << showbase << hex << value << endl;**	**0x40**
noshowbase	Не выводить индикатор основания системы счисления (по умолчанию).	**int value = 64; cout << noshowbase << hex << value << endl;**	**40**

Манипуляторы в C++

Манипулятор	Назначение	Пример	Результат
uppercase	В шестнадцатеричной системе счисления использовать буквы верхнего регистра (по умолчанию установлены буквы нижнего регистра).	**int value = 255; cout << uppercase << hex << value << endl;**	**FF_{16}**
nouppercase	В шестнадцатеричной системе счисления использовать буквы нижнего регистра (по умолчанию).	**int value = 255; cout << nouppercase << hex << value << endl;**	**ff_{16}**
showpos	Вывод знака плюс + для положительных чисел	**int value = 255; cout << showpos<< value << endl;**	**+255**
noshowpos	Не выводить знак плюс **+** для положительных чисел (по умолчанию).	**int value = 255; cout <<noshowpos<< value << endl;**	**255**

Манипуляторы в C++

Манипулятор	Назначение	Пример	Результат
scientific	Вывод чисел с плавающей точкой в экспоненциальной форме	**double value = 1024.165; cout << scientific << value << endl;**	**1.024165e+003**
fixed	Вывод чисел с плавающей точкой в фиксированной форме (по умолчанию).	**double value = 1024.165; cout << fixed << value << endl;**	**1024.165**
setw(int number)	Установить ширину поля, где number — количество позиций, символов (выравнивание по умолчанию по правой границе). Манипулятор с параметром.	**cout << setw(40) << «cppstudio.com» << endl;**	**__cppstudio.com**
right	Выравнивание по правой границе(по умолчанию). Сначала необходимо установить ширину поля(ширина поля должна быть заведомо большей чем, длина выводимой строки).	**cout << setw(40) << right << «cppstudio.com» << endl;**	**__cppstudio.com**

Манипуляторы в C++

Манипулятор	Назначение	Пример	Результат
left	Выравнивание по левой границе. Сначала необходимо установить ширину поля(ширина поля должна быть заведомо большей чем, длина выводимой строки).	**cout << setw(40) << left << «cppstudio.com» << endl;**	**cppstudio.com__**
setprecision(int count)	Задаёт количество знаков после запятой, где count — количество знаков после десятичной точки	**cout << fixed << setprecision(3) << (13.5 / 2) << endl;**	**6.750**
setfill(int symbol)	Установить символ заполнитель. Если ширина поля больше, чем выводимая величина, то свободные места поля будут наполняться символом symbol — символ заполнитель	**cout << setfill('0') << setw(4) << 15 << ends << endl;**	**0015**

Строки

- В стандартной библиотеке имеется тип ***string*.**
- Тип ***string*** обеспечивает множество полезных операций над строками
- Заголовочный файл **<string>.**

Строки

- Для типа string доступны операторы сравнения ==, <, > , <= , >=, !=
- Доступны операции +, +=. Данные операторы реализуют операцию конкатенации.
- Операцией взятия индекса [ ] str[10]
- Метод at() предлагает похожую схему доступа. метод at(), обеспечивает проверку границ и генерирует сключение out_of_range

```
str.at(7)
```

Методы string

size Возвращает длину строки

length Возвращает длину строки

max_size Возвращает максимальный размер строки

resize Изменяет размер строки

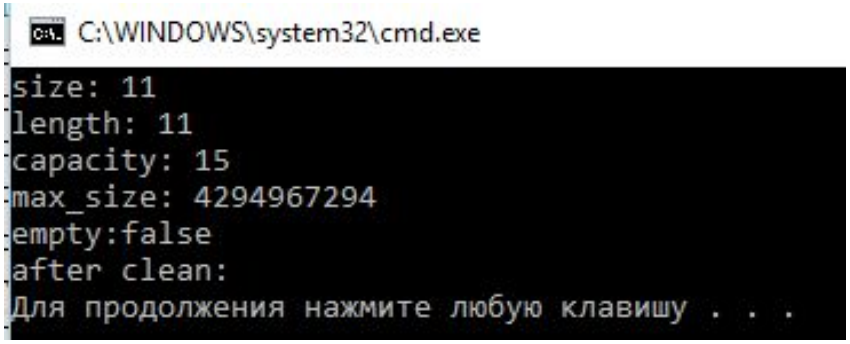
capacity возвращет размер выделенной памяти

clear очищает строку

empty проверка на пустоту строки.

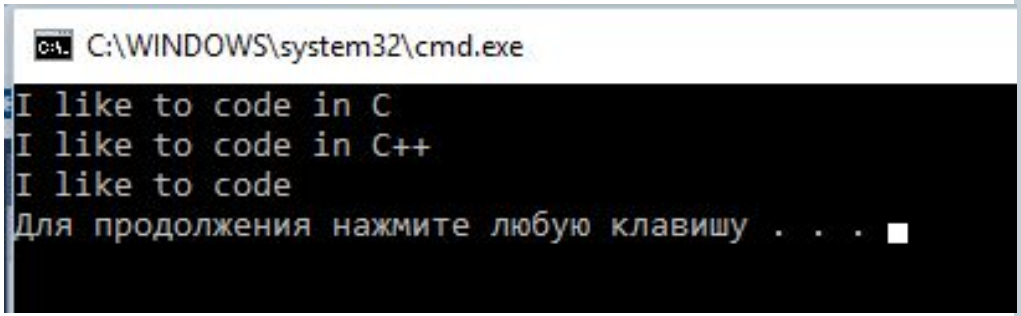
Методы string


```cpp
#include <iostream>
#include <string>

int main()
{
    std::string str("Test string");
    std::cout << "size: " << str.size() << std:: endl;
    std::cout << "length: " << str.length() << std:: endl;
    std::cout << "capacity: " << str.capacity() << std:: endl;
    std::cout << "max_size: " << str.max_size() << std:: endl;
    std::cout << std::boolalpha << "empty:" << str.empty() << std::endl;
    str.clear();
    std::cout << "after clean:" << str << std::endl;
    return 0;
}
```



Методы string


```
#include <iostream>
#include <string>

int main()
{
    std::string str("I like to code in C");
    std::cout << str << '\n';
    std::string::size_type sz = str.size();
    str.resize(sz + 2, '+');
    std::cout << str << '\n';
    str.resize(14);
    std::cout << str << '\n';
    return 0;
}
```



Методы string

insert вставляет в строку

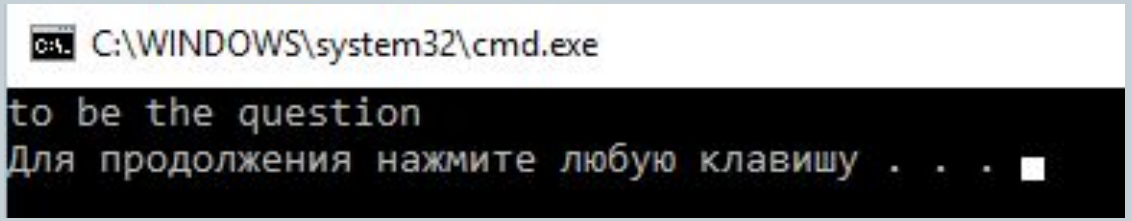
erase удаляет последовательность символов из строки

replace заменяет часть строки

Методы string

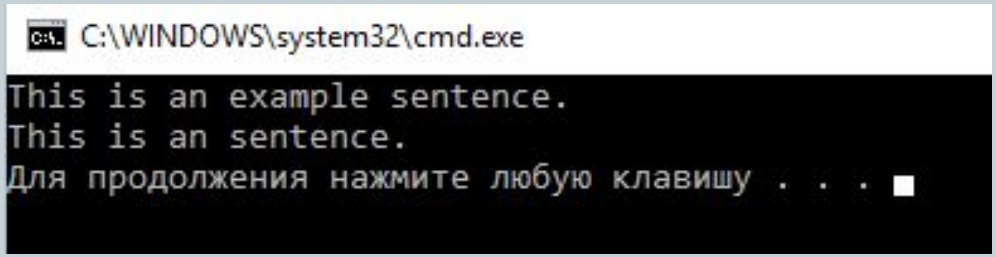
- #include <iostream>
- #include <string>

```
int main()
{
    std::string str = "to be question";
    std::string str2 = "the ";
// used in the same order as described above:
    str.insert(6, str2);   // to be (the )question
    std::cout << str << '\n';
    return 0;
}
```



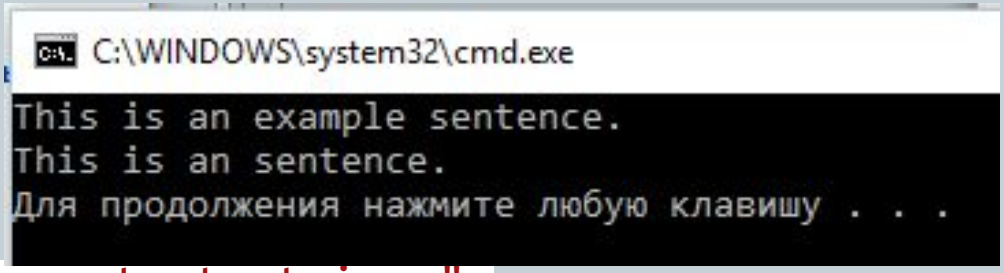
Методы string


```
#include <iostream>
#include <string>

int main()
{
  std::string str("This is an example sentence.");
  std::cout << str << std::endl;
  str.erase(10, 8);
  std::cout << str << std::endl;
  return 0;
}
```



Методы string


```
#include <iostream>
#include <string>

int main()
{
  std::string base = "this is a test string.";
  std::string str2 = "n example";
  std::string str = base;                 // "this is a test string."
  str.replace(9, 5, str2);                // "this is an example
string." (1)
  std::cout << base << std::endl;
  std::cout << str << std::endl;
  return 0;
}
```

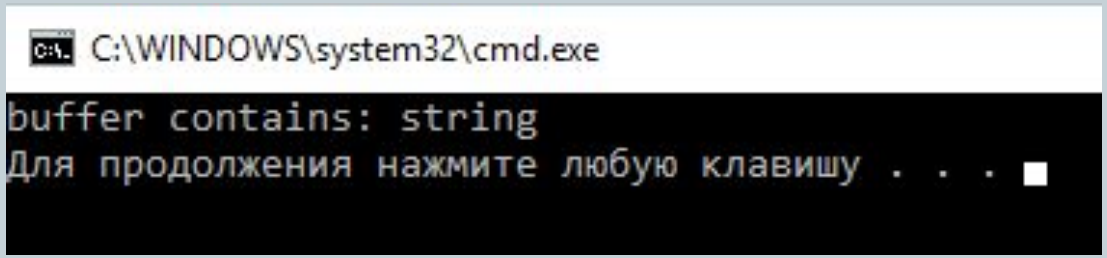


Методы string

- copy копирование последовательности символов из строки
- find поиск первого вхождения подстроки в строку
- rfind поиск последнего вхождения подстроки в строку
- substr возвращает подстроку строки

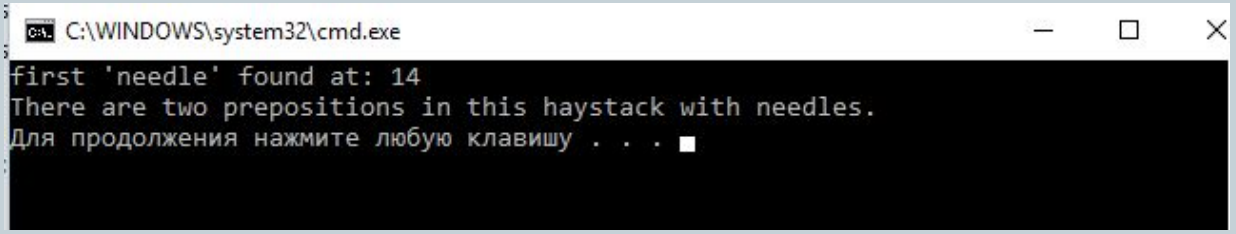
Методы string


```
#include <iostream>
#include <string>

int main()
{
  char buffer[20];
  std::string str("Test string...");
  std::size_t length = str.copy(buffer, 6, 5);
  buffer[length] = '\0';
  std::cout << "buffer contains: " << buffer << '\n';
  return 0;
}
```



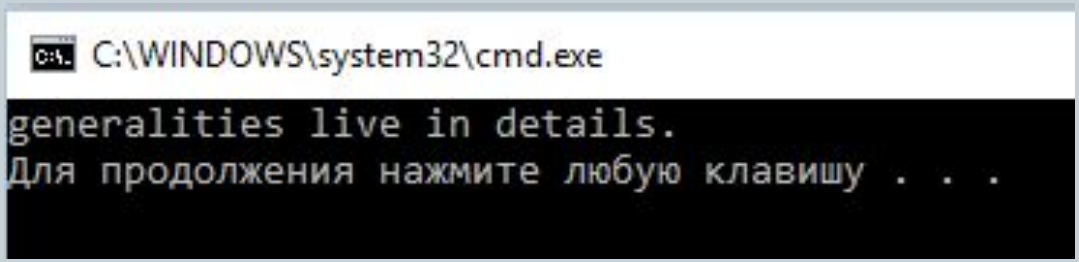
Методы string


```
#include <iostream>
#include <string>

int main()
{
   std::string str("There are two needles in this haystack with needles.");
   std::string str2("needle");
   std::string::size_type found = str.find(str2);
   if (found != std::string::npos)
     std::cout << "first 'needle' found at: " << found << std::endl;

   str.replace(str.find(str2), str2.length(), "preposition");
   std::cout << str << std::endl;

   return 0;
}
```



Методы string


```
#include <iostream>
#include <string>

int main()
{
    std::string str = "We think in generalities, but we live in details.";
    std::string str2 = str.substr(12, 12);
    std::string::size_type pos = str.find("live");
    std::string str3 = str.substr(pos);
    std::cout << str2 << ' ' << str3 << '\n';
    return 0;
}
```



Контейнеры

- Многие вычисления подразумевают создание наборов объектов в различных формах и обработку таких наборов.
- Класс, главной целью которого является хранение объектов, называется ***контейнером.***
- Реализация контейнеров, подходящих для данной задачи, и поддержка их основными полезными операциями — важнейшие шаги при написании любой программы.

Вектор

- Встроенные массивы имеют фиксированный размер.
  - Если мы выберем слишком большой размер, то впустую израсходуем память.
  - Если же выбранный размер слишком мал, массив может переполниться.
- В любом случае нам придется написать код низкого уровня для управления памятью.
- Стандартная библиотека предоставляет вектора (***vector***), которые сами позаботятся об этом

Вектор


```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
struct Entry {
   string name;
   int number;
};
vector<Entry> phone_book(1000);

void print_entry(int i) {
   cout<<phone_book[i].name << phone_book[i].number << endl;
}

void addjentries(int n){
   phone_book.resize(phone_book.size() + n);
}
```