

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

1

Стандартная библиотека

Введение

2

- 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

3

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

Ввод/вывод

4

- 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

5

- Доступ к функциям осуществляется через операцию **точка**, а в круглых скобках 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

6

- Флаги вывода объявлены в классе `ios`

- Установка флага

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

- Снятие флага

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

Флаги

7

- 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++

8

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

Флаги форматирования в C++

9

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

Флаги форматирования в C++

10

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

Манипуляторы в C++

11

- 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++

12

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

Манипуляторы в C++

13

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

Манипуляторы в C++

14

Манипулятор	Назначение	Пример	Результат
uppercase	В шестнадцатеричной системе счисления использовать буквы верхнего регистра (по умолчанию установлены буквы нижнего регистра).	<pre>int value = 255; cout << uppercase << hex << value << endl;</pre>	FF₁₆
nouppercase	В шестнадцатеричной системе счисления использовать буквы нижнего регистра (по умолчанию).	<pre>int value = 255; cout << nouppercase << hex << value << endl;</pre>	ff₁₆
showpos	Вывод знака плюс + для положительных чисел	<pre>int value = 255; cout << showpos << value << endl;</pre>	+255
noshowpos	Не выводить знак плюс + для положительных чисел (по умолчанию).	<pre>int value = 255; cout << noshowpos << value << endl;</pre>	255

Манипуляторы в C++

15

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

Манипуляторы в C++

16

Манипулятор	Назначение	Пример	Результат
left	Выравнивание по левой границе. Сначала необходимо установить ширину поля(ширина поля должна быть заведомо большей чем, длина выводимой строки).	<pre>cout << setw(40) << left << «cppstudio.com» << endl;</pre>	cppstudio.com__
setprecision(int count)	Задаёт количество знаков после запятой, где count — количество знаков после десятичной точки	<pre>cout << fixed << setprecision(3) << (13.5 / 2) << endl;</pre>	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	<pre>cout << setfill('0') << setw(4) << 15 << ends << endl;</pre>	0015

Строки

17

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

Строки

18

- 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

19

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

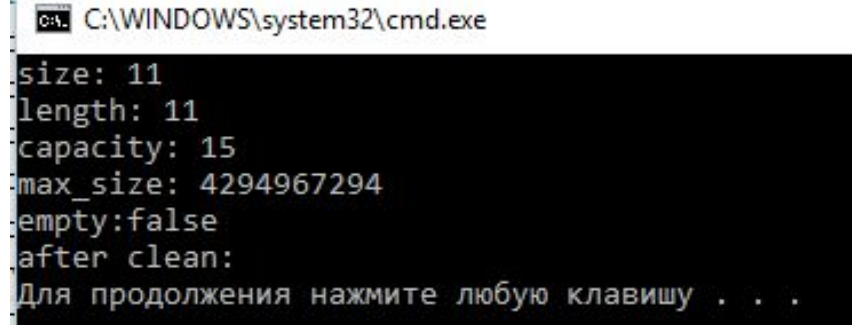
empty проверка на пустоту строки.

Методы string

20

```
#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;
}
```



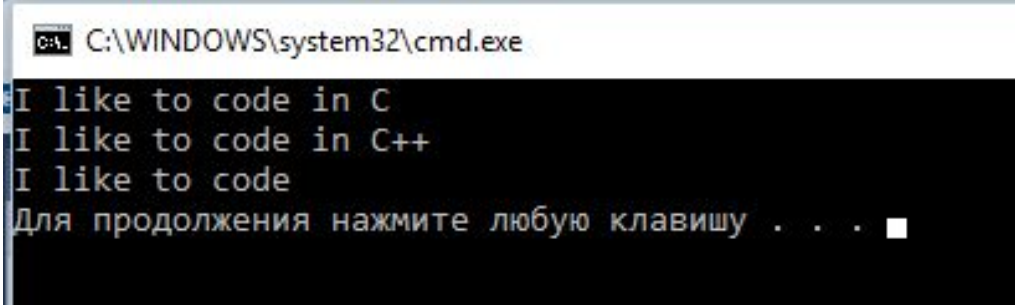
```
C:\WINDOWS\system32\cmd.exe
size: 11
length: 11
capacity: 15
max_size: 4294967294
empty:false
after clean: . . . .
Для продолжения нажмите любую клавишу . . .
```

Методы string

21

```
#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;
}
```



C:\WINDOWS\system32\cmd.exe

```
I like to code in C
I like to code in C++
I like to code
Для продолжения нажмите любую клавишу . . .
```

Методы string

22

insert вставляет в строку

erase удаляет последовательность символов
из строки

replace заменяет часть строки

Методы string

23

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

```
int main()
```

```
{
```

```
    std::string str = "to be question";
```

```
    std::string str2 = "the ";
```

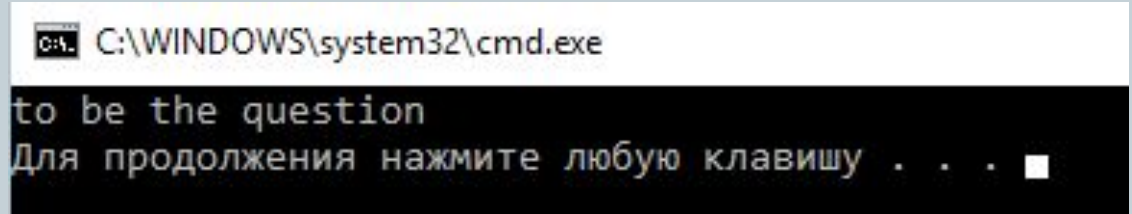
```
// used in the same order as described above:
```

```
    str.insert(6, str2);    // to be (the )question
```

```
    std::cout << str << '\n';
```

```
    return 0;
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
to be the question
Для продолжения нажмите любую клавишу . . .
```

Методы string

24

```
#include <iostream>
```

```
#include <string>
```

```
int main()
```

```
{
```

```
    std::string str("This is an example sentence.");
```

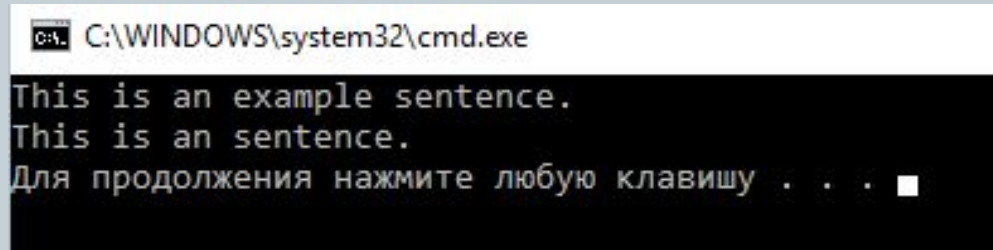
```
    std::cout << str << std::endl;
```

```
    str.erase(10, 8);
```

```
    std::cout << str << std::endl;
```

```
    return 0;
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
This is an example sentence.
This is an sentence.
Для продолжения нажмите любую клавишу . . .
```


Методы string

25

```
#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."
```

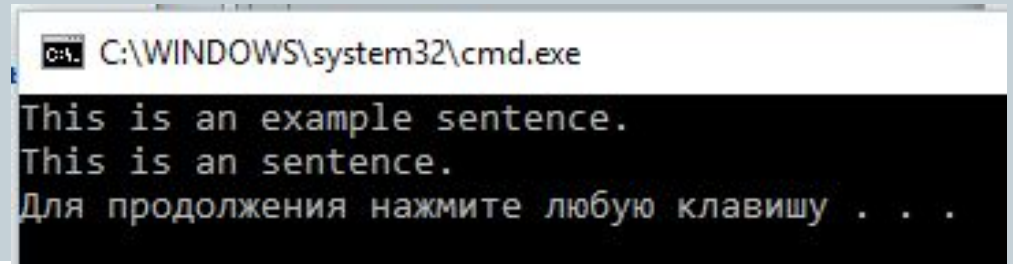
```
    str.replace(9, 5, str2);          // "this is an example
string." (1)
```

```
    std::cout << base << std::endl;
```

```
    std::cout << str << std::endl;
```

```
    return 0;
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
This is an example sentence.
This is an sentence.
Для продолжения нажмите любую клавишу . . .
```

Методы string

26

- **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

27

```
#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;
```

```
}
```

```
C:\WINDOWS\system32\cmd.exe
```

```
buffer contains: string
```

```
Для продолжения нажмите любую клавишу . . .
```

Методы string

28

```
#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)
```

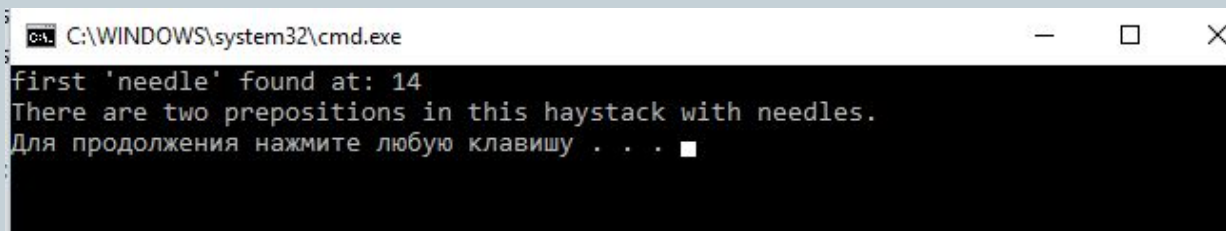
```
        std::cout << "first 'needle' found at: " << found << std::endl;
```

```
    str.replace(str.find(str2), str2.length(), "preposition");
```

```
    std::cout << str << std::endl;
```

```
    return 0;
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe
first 'needle' found at: 14
There are two prepositions in this haystack with needles.
Для продолжения нажмите любую клавишу . . .
```

Методы string

29

```
#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;
```

```
}
```

```
C:\WINDOWS\system32\cmd.exe
```

```
generalities live in details.
```

```
Для продолжения нажмите любую клавишу . . .
```

Контейнеры

30

- 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

31

- 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

32

```
#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);
}
```