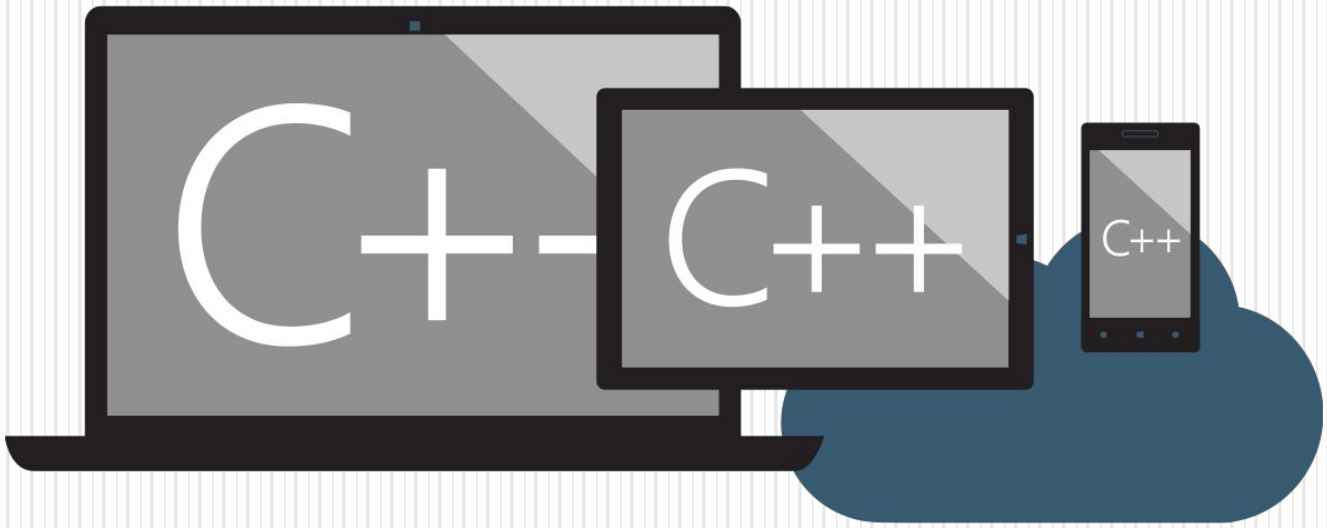
Лекция 7

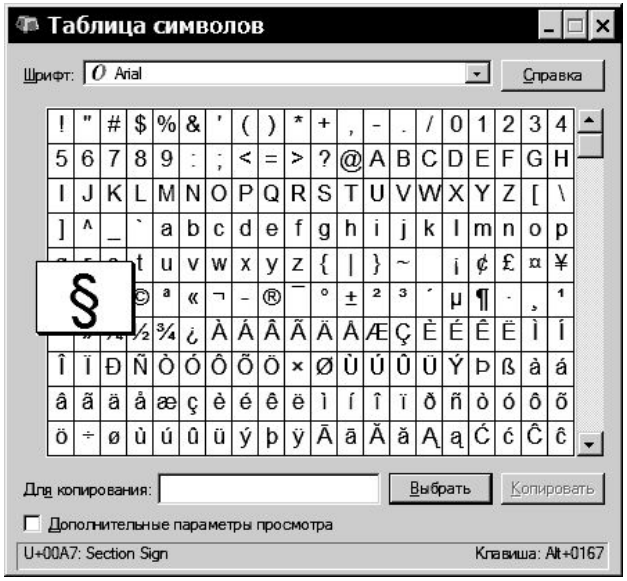
Строки. Работа со строками.



Содержание

Введение

Кроме числовой информации компьютер может обрабатывать символьную информацию. Для представления текстовой информации в языке C++ используются символы (символьные константы), символьные переменные и строки (строковые константы).



Символ

Символ (символьная константа) – это буква, цифра или любой другой изображаемый знак, заключенный в одинарные кавычки (апострофы).

Например: **‘F’ ‘x’ ‘7’** ‘+’ ‘>’ **‘;’**

Для хранения и обработки символов используют переменные типа char. Значением переменной char может быть любой символ.

Переменные символьного типа должны быть описаны следующим образом:

char список имен переменных;

Например: **char a, z=’*’;**

Переменную символьного типа можно сравнивать с другой переменной типа char. Сравнение основано на том, что каждому символу поставлено в соответствие число (внутренний код), причем символу ‘0’ соответствует число меньшее, чем символу ‘9’, символу ‘A’ – меньшее, чем ‘B’, символу ‘Z’ – меньшее, чем ‘a’. Таким образом, можно записать:

‘0’<’1’<...<’9’<...<’A’<...<’B’<...<’Z’<’a’<’b’<...’z’

Строка

Строка (строковая константа) - это последовательность символов, заключенная в двойные кавычки.

Например:

“1234567890”

“\t Строковая константа”

“ Начало строки \n и конец строки”

программе любая строка объявляется как массив символов:

char mm[]=”массив символов” ;

Строка представляет собой массив символов, заканчивающийся нуль-символом. При объявлении в программе строковой константы транслятор C++ автоматически добавляет в конец строки NULL символ (\0), поэтому длина строковой константы на 1 больше, чем количество символов в ней.

Строка

Нуль-символ - это символ с кодом, равным 0, что записывается в виде управляющей последовательности '\0'. По положению нуль-символа определяется фактическая длина строки. Строку можно инициализировать строковым литералом:

char str[10] = "Vasia";

// выделено 10 элементов с номерами от 0 до 9

// первые элементы - 'V', 'a', 's', 'i', 'a', '\0'

В данном примере под строку выделяется 10 байт, 5 из которых занято под символы строки, а шестой - под нуль-символ.

Если строка при определении инициализируется, ее размерность можно опускать (компилятор сам выделит соответствующее количество байт):

char str[] = "Vasia"; // выделено и заполнено 6 байт

Знак равенства перед строковым литералом означает инициализацию, а не присваивание.

Максимальная длина строки – 256 символов. В записи строки может быть и один символ:

char st[]= ”A”; /* строка А */

char ss=’A’ ; /* символ А */

Для ввода и вывода строк можно использовать объекты cin и cout.

```
#include "pch.h"
#include <iostream>
using namespace
std;
int main() {
    const int n = 80;
    char s[n];
    cin >>s; cout << s
<< endl;
 return 0; }
```

Можно реализовать ввод слов входной строки в отдельные строковые переменные.

```
#include "pch.h"
#include <iostream>
using namespace
std;
int main(){
    const int n = 80;
    char s[n], t[n], r[n];
    cin >> s >> t >> r;
   cout << s << endl
<< t << endl << r <<
endl;
return 0; }
```

Getline и get

Можно также реализовать ввод строки, состоящей из нескольких слов, в одну строковую переменную. Для этого используются методы getline или get класса iostream, объектом которого является cin.

Метод getline считывает из входного потока n-1 символов или менее (если символ перевода строки встретится раньше) и записывает их в строковую переменную s. Символ перевода строки также считывается (удаляется), но не записывается в строковую переменную, вместо него размещается завершающий 0.

Метод get работает аналогично, но оставляет в потоке символ перевода строки. В строковую переменную добавляется завершающий 0.

```
#include "pch.h"
#include <iostream>
using namespace std;
int main() {
const int n = 80;
char s[n];
cin.getline(s, n);
cout << s << endl;
cin.get (s, n);
cout << s << endl;
return 0; }
```


Пример работы со строками

```
#include "pch.h"
#include <iostream>
using namespace std;
 int main() {
  char str[80];   /* Зарезервировали место для строки */
  cout<<"Введите строку длиной менее 80 символов:"<<endl;
  cin>>str;  /* читает строку с клавиатуры, пока не нажмете клавишу
                                        Enter */
  cout<<"Вы ввели строку \n" << str;
  printf("Введите строку длиной менее 80 символов:");
  gets(str);
   /* читает строку с клавиатуры, пока не нажмете клавишу Enter */
  puts(str);
  printf("ВВедите еще одну строку длиной менее 80 символов: ");
  scanf("%s", str);
  /* читает строку с клавиатуры, пока не встретится пробел */
printf("Вы ввели строку %s \n", str);
return 0;
}
```


Функции работы со строками

1) strlen (str) – возвращает фактическую длину строки str, не включая нуль-символ.

2) strcpy (str1, str2) – копирует символы из строки str2 в строку str1.

3) strncpy (str1, str2, n) – копирует n символов из строки str2 в строку str1.

4) strcat (str1, str2) – присоединяет строку str2 к строке str1 (конкатенация).

5) strncat (str1, str2, n) – добавляет в конец строки str1 n первых символов из строки str2.

6) strrev (str) – возвращает строку str, в которой изменен порядок следования символов на обратный.

7) strcmp (str1, str2) – Возвращает 0, если str1=str2, возвращает <0 если str1<str2, >0 если str1>str2.

8) strncmp (str1, str2, n) – возвращает 0, если str1=str2, возвращает <0 если str1<s, >0, если str1>str2 сравниваются только первые maxlen символов двух строк.

9) stricmp(str1, str2) – возвращает 0, если str1= str2, возвращает <0, если str 1< str2, и возвращает >0, если str1>0, не проверяется регистр букв.

Функции работы со строками

10) strnicmp (str1, str2, n) – Возвращает 0, если str1=str2, возвращает <0, если str1<str2, и возвращает >0, если str1<str2, сравниваются только n символов двух строк. Не проверяется регистр букв.

11) strstr (str, pst) – возвращает индекс первого символа в строке str, с которого начинается подстрока pst.

12) strupr (str) – преобразует буквы нижнего регистра в строке str в буквы верхнего регистра.

13) strlwr (str) – преобразует буквы верхнего регистра в строке str в буквы нижнего регистра.

14) atoi (smb) – преобразует символьное представление числа smb в целое число.

15) atof(smb) – преобразует символьное представление числа smb в вещественное число.

16) strchr (source, ch) - gоиск в строке source первого вхождения символа ch;

17) spirntf выводит в строку, адрес которой задается параметром.

sprintf (dest, format,.....);

18) Функция sscanf читает из строки, адрес которой задается первым параметром.

sscanf (dest, format,....);

Пример использования функций spirntf и sscanf

```
#include "pch.h"
#include <iostream>
using namespace std;
char destination[ 100];
int main () {
int i=-56;
float r=124.96752;
sprintf( destination, "%d", i );
printf( "\n destation=%s", destation);
sprintf( destation, "%.3f ", r);
printf ("\n destation =%s" , destation);
sprintf (destation, "i=%d, r= %f", i, r);
printf ("\ n destation= %s \n", destation);
return 0;
}
```


В заголовочных файлах <stdlib.h> и <cstdlib> также содержатся полезные функции преобразования строк в числа.

double atof (const char* p) преобразует переданную строку в double;

int atoi (const char* p) преобразует переданную строку в int;

long atol (const char* p) преобразует переданную строку в long.

Пример программы заполнения массива типа double из строки:

```
#include "pch.h"
#include <iostream>
#include <string>
#include <stdlib>
using namespace std;
int main(){
  char s[] = "2, 38.5, 70,  0, 0,  1",
*p = s;
  double m[10];
  int i = 0;
  do {
          m[i++] = atof(p);   if (i>9)
break;
  } while (p = strchr(p,   ','), p++);
  for ( int k = 0; k<i; k++)
        printf("%5.2f ", m[k]);
  return 0;
}
```


Для работы с символами в стандартной библиотеке (заголовочные файлы <ctype. h> и <cctype>) используются функции, представленные в таблице

Имя	Проверка на принадлежность символа множеству
isalnum	букв и цифр (A-Z, a-z, 0-9)
isalfa	букв (A-Z, a-z)
iscntrl	управляющих символов (с кодами 0..31 и 127)
isdigit	цифр (0-9)
jsgraph	печатаемых символов, кроме пробела (isalfa | isdigit | ispunct)
islower	букв нижнего регистра (a-z)
isprint	печатаемых символов
ispunct	знаков пунктуации
isspace	символов-разделителей
isupper	букв верхнего регистра (A-Z)
isxdigit	шестнадцатеричных цифр (A-F, a-f, 0-9)

Указатели и строки

При работе со строками часто используются указатели.

Пример

char *str = "Vasia"

создается не строковая переменная, а указатель на строковую константу, изменить которую невозможно.

Для размещения строк в динамической памяти можно использовать два варианта:

1 описать указатель на char;

2 выделить память с помощью new или malloc.

char *p = new char [m];

char *q = (char *) malloc (m * sizeof(char));

Динамические строки

Динамические строки, как и другие динамические массивы, нельзя инициализировать при создании.

Оператор **char *str = "Скоро лето и сессия!!!”**

создает не строковую переменную, а указатель на строковую константу, изменить которую невозможно.

Для демонстрации работы с указателями рассмотрен пример сравнивания строк src и строки dest. Алгоритм имеет вид:

```
char src[10], dest[10];
...
for (int i = 0; i<=strlen(src); i++) dest[i] = src[i];
```

Длина строки определяется с помощью функции strlen, которая вычисляет длину, выполняя поиск нуль-символа. Таким образом, строка фактически просматривается дважды. Более эффективным будет использовать проверку на нуль-символ непосредственно в программе. Увеличение индекса можно заменить инкрементом указателей (для этого память под строку src должна выделяться динамически, а также требуется определить дополнительный указатель и инициализировать его адресом начала строки dest). Пример представлен на следующем слайде.

В цикле производится посимвольное присваивание элементов строк с одновременной инкрементацией указателей. Результат операции присваивания - передаваемое значение, которое, собственно, и проверяется в условии цикла, поэтому можно поставить присваивание на место условия, а проверку на неравенство нулю опустить (при этом завершающий нуль копируется в цикле, и отдельного оператора для его присваивания не требуется). В результате цикл копирования строки принимает вид: **while (*d++ = *src++);**

```
#include "pch.h"
#include <iostream>
#include <string>
#include <conio>
using namespace std;
int main(){
char *src = new char [10];
char *dest = new char [10], *d = dest;
cin >> src;
while ( *src != 0) *d++ = *src++;
*d = 0; // завершающий нуль
cout << dest;
return 0;
}
```


Примеры работы со строками

Необходимо ввести строку X с клавиатуры. Переписать все символы данной строки X в новую строку Y в обратном порядке.

```
#include "pch.h"
#include <iostream>
#include <string>
#include <conio>
main ()
{char X[256], Y[256];
clrscr(); puts ("\n     введите строку X: "); gets (X);
strcpy(Y, X) ; /* строка X копируется в строку Y */ strrev(Y) ; /* строка Y переписывается в обратном порядке*/ puts("\nСтрока Y: “);
puts(Y);
}
```


Примеры работы со строками

Пример ввода с клавиатуры строки St и подсчета в ней, сколько раз встретилась буква ‘a’.

```
#include "pch.h"
#include <iostream>
#include <string>
using namespace std;
int main ()
{char St[80];
int ka, i=0;
clrscr();
printf ("\n Введите строку с точкой \n");
gets (St);
for (ka=0,i=0;St[i]!='.';i++)
if (St[i]=='a') ka++; printf("\n ka=%d ",ka);}
```


Примеры работы со строками

Далее представлен пример записи всех слов введенной строки A в одномерный строковый массив B.

```
#include "pch.h"
#include <iostream>
#include <conio>
#include <string>
int main ()
{
char a[80], b[20][20];
int i, j, str, stb;
clrscr();
printf ("\n введите предложение \n"); gets(a);
for (i=0, str=stb=0; i<strlen(a);i++)
if (a[i]!=' ') {b[str][stb]=a[i]; stb++;}
else {b[str][stb]='\0';str++;stb=0;}          b[str][stb]='\0';
printf ("\n Строковый массив  \n");
for (i=0;i<=str; i++) puts(b[i]);
getch();
}
```


Примеры работы со строками

Далее приведен пример программы, которая запрашивает пароль не более трех раз.

```
#include "pch.h"
#include <iostream>
#include <string>
using namespace std;
int main(){
char s[5], passw[] = "kuku"; // passw – эталонный пароль.
// Можно описать как *passw = "kuku";
int i, k = 0;
for (i = 0; !k && i<3; i++){
printf("\nвведите пароль:\n");
gets(s); // функция ввода строки
if (strstr(s,passw))k = 1; // функция сравнения строк
}
if (k) printf("\n пароль принят");
else printf("\n пароль не принят");
return 0;
}
```

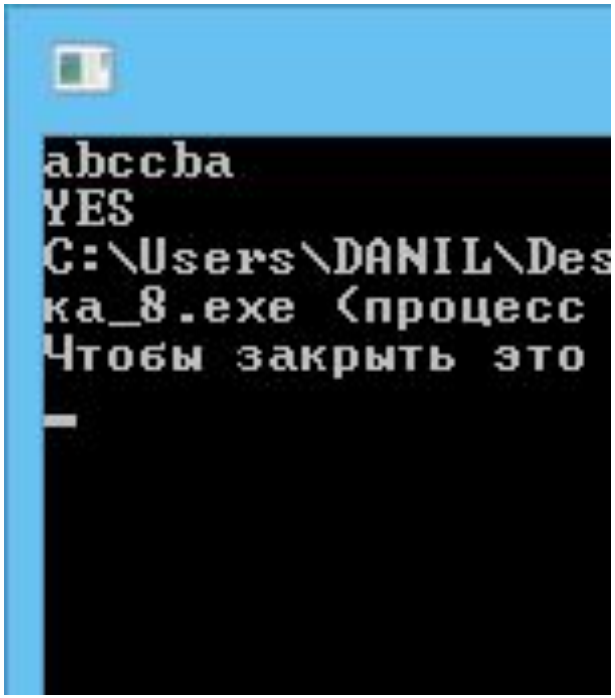

Примеры работы со строками

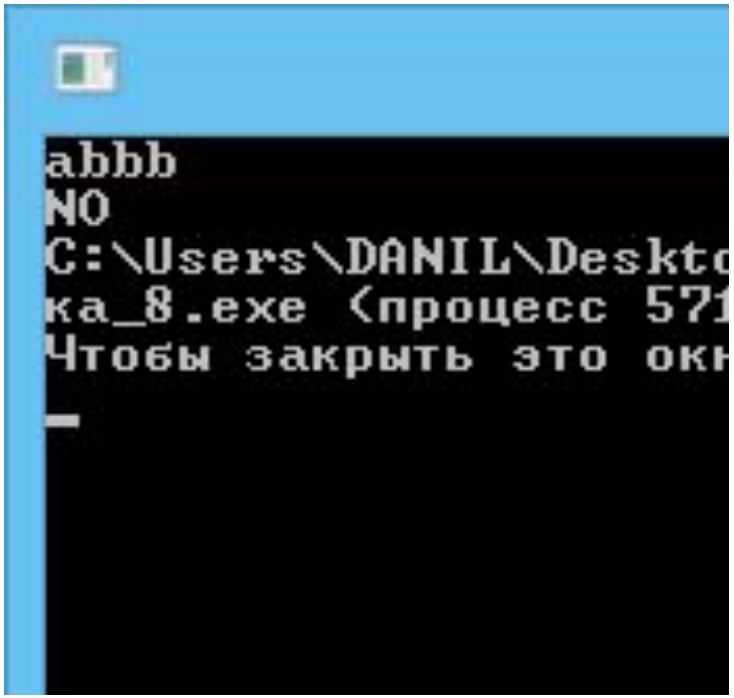
В следующем примере необходимо проверить является ли строка палиндромом. Палиндром – это выражение, которое читается одинаково слева направо и справа налево.

Для реализации данного кода необходимо считать строку без пробелов с помощью функции getline(). Затем следует выполнить проход до половины строки (не зависит от четности элементов в строке) и проверить элементы строки с номерами 0 и n-1, 1 и n-2 и т.д. Если будет хотя бы одно не совпадение, то программа выведет «NO» и завершит работу. Иначе «YES».
Реализация данного примера приведена на следующем слайде.

```
#include "pch.h"
#include <iostream>
#include <string>
using namespace std;
int main()
{
        string s;
 getline(cin,s);
        for (int i = 0; i < (s.length() / 2);
i++)
                if (s[i] != s[s.length() - 1 - i])
{
        cout << "NO";
 return 0;
}
 cout << "YES";
 return 0;
}
```


Результаты выполнения программы



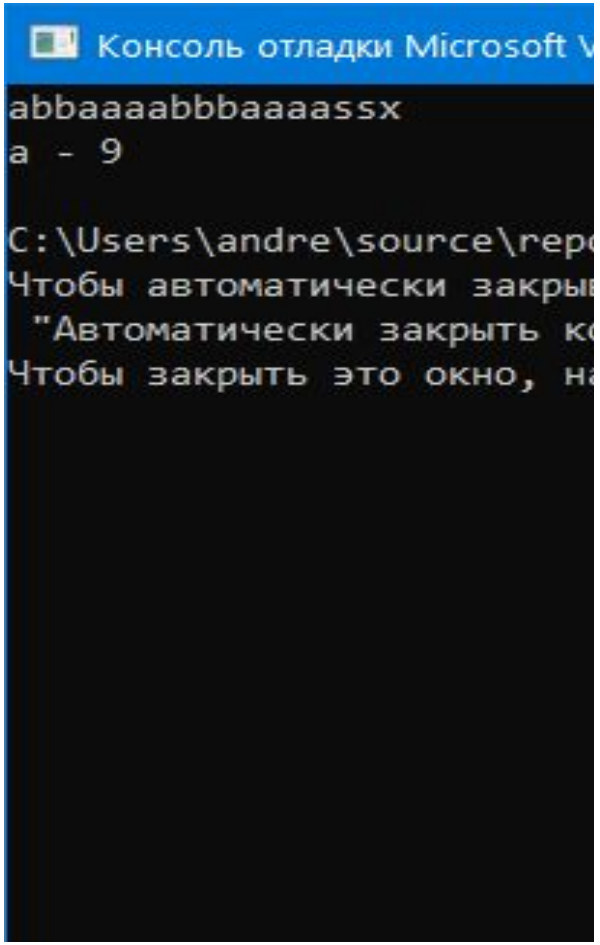


Примеры работы со строками

Далее редставлен пример определяющий какая буква в строке встречается больше других.

```
#include "pch.h"
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;
int main()
{
	string s;
	char p = 'a';
	getline(cin, s);
	vector<pair <int, char>> x(26);
	for (int i = 0; i < 26; i++) {
		x[i].second = p + i;
	}
	for (int i = 0; i < s.length(); i++) {
		int j = 0;
		while (j < 26) {
			if (s[i] == x[j].second) {
				x[j].first++;
				break;
			}
			else j++;
		}
	}
	sort(x.rbegin(), x.rend());
	cout << x[0].second<<" - "<<x[0].first << endl;
	return 0;
}
```


Сначала задается массив пар, где первый элемент пары количество повторений буквы в строке, второй элемент символ который встречается в строке. Далее считывается строка без пробелов с помощью **getline()**. Выполняется проход по строке, в цикле сравнивается элемент массива с символом строки, если элементы совпадают, то первый элемент пары увеличивается и происходит выход из цикла сравнения, и продолжается проход по строке. После работы цикла массив пар сортируется по количеству элементов в обратном порядке, т.к. нужно вывести букву с наибольшей частотой встречаемости.



Код программы

Контрольные вопросы

1. Что представляет собой строка?
2. Перечислите способы ввода-вывода строк.
3. Назовите некоторые функции работы со строками.
4. Как реализуется работы указателей и строк?

Список литературы

- Павловская Т.А. С/С++. Программирование на языке высокого уровня / Т. А. Павловская. - СПб.: Питер, 2004. - 461 с.: ил.
- Павловская Т.А. С/С ++. Структурное программирование: Практикум / Т.А. Павловская, Ю.А. Щупак. СПб.: Питер, 2007. - 239 с.: ил.
- Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование: Практикум. - СПб.: Питер, 2006. - 265 с: ил.
- Кольцов Д.М. 100 примеров на Си. - СПб.: “Наука и техника”, 2017 - 256 с.
- 5 Доусон М. Изучаем С++ через программирование игр. - СПб.: “Питер”, 2016. - 352.
- Седжвик Р. Фундаментальные алгоритмы на С++. Анализ/Структуры данных/Сортировка/Поиск: Пер. с англ. Роберт Седжвик. - К.: Издательство “Диасофт”, 2001. - 688с.
- Сиддкхартха Р. Освой самостоятельно С++ за 21 день. - М.: SAMS, 2013. - 651 с.
- Стивен, П. Язык программирования С++. Лекции и упражнения, 6-е изд. Пер. с англ. - М.: ООО "И.Д. Вильямс", 2012. - 1248 с.
- Черносвитов, А. Visual C++: руководство по практическому изучению / А. Черносвитов . - СПб. : Питер, 2002. - 528 с. : ил.

Список литературы

- Страуструп Б. Дизайн и эволюция языка C++. - М.: ДМК, 2000. - 448 с.
- Мейерс С. Эффективное использование C++. - М.: ДМК, 2000. - 240 с.
- Бадд Т. Объектно-ориентированное программирование в действии. - СПб: Питер, 1997. - 464 с.
- Лаптев В.В. С ++. Объектно-ориентированное программирование: Учебное пособие.- СПб.: Питер, 2008. - 464 с.: ил.
- Страуструп Б. Язык программирования C++. Режим доступа: http://8361.ru/6sem/books/Straustrup-Yazyk_programmirovaniya_c.pdf.
- Керниган Б., Ритчи Д. Язык программирования Си. Режим доступа: http://cpp.com.ru/kr_cbook/index.html.
- Герберт Шилдт: C++ базовый курс. Режим доступа: https://www.bsuir.by/m/12_100229_1_98220.pdf,
- Богуславский А.А., Соколов С.М. Основы программирования на языке Си++. Режим доступа: http://www.ict.edu.ru/ft/004246/cpp_p1.pdf.
- Линский, Е. Основы C++. Режим доступа: https://www.lektorium.tv/lecture/13373.
- Конова Е. А., Поллак Г. А. Алгоритмы и программы. Язык C++: Учебное пособие. Режим доступа: https://vk.com/doc7608079_489807856?hash=e279524206b2efd567&dl=f85cf2703018eeaa2