Лекция 7. Использование потоков для работы с памятью

Область памяти может рассматриваться как поток, который ассоциируется со строкой из символов некоторой длины.

**strstream.lib**

**strstream** - ввод-вывод.

**istrstream** - ввод (чтение из строки).

**ostrstream** - вывод (запись в строку).

**Класс strstream**:

strstream();

strstream(char\* строка, int количество\_символов, int режим\_открытия);

**Класс istrstream**:

istrstream(char\* строка);

istrstream(char\* строка, int количество\_символов);

Первый конструктор позволяет рассматривать всю строку buf как входной поток, ограниченный нулевым символом '\0' конца строки, а второй делает то же самое, но разрешает использовать только n символов строки.

**Класс ostrstream**:

ostrstream();

ostrstream(char\* строка, int количество\_символов, int режим\_открытия);

Первый конструктор создает динамический выходной строковый поток, память которому выделяется по мере необходимости. Второй конструктор определяет строку как выходной поток в режиме открытия, ограниченный n символами.

Потоки наследуют операции форматированного и неформатированного ввода-вывода.

Пример. Ввести данные различных типов и вывести в виде строки.

#include <iostream>

#include <strstream>

using namespace std;

int main()

{

char ch; int x;

char str[30], buf1[50], buf2[50];

cout<<"Input ch, x, str: ";

cin.getline(buf1, 50);

istrstream ist(buf1, strlen(buf1));

ist>>ch>>x>>str;

ostrstream ost(buf2, 50);

ost<<"ch="<<ch<<endl<<"x="<<x<<endl<<"str="<<str<<endl<<endl;

cout<<buf2;

return 0;

}

В отличие от буферизации данных системой ввода/вывода в данном случае имеется возможность изменять содержимое буфера. Использованные выше классы в настоящее время считаются устаревшими. В новых версиях языка С++ предпочтительно использовать библиотеку sstream и соответствующие классы: **stringstream, istringstream, ostringstream**.

Пример. Преобразовать строковые данные в числовые.

#include <iostream>

#include <sstream>

using namespace std;

int main()

{

int k;

double x, s;

stringstream st;

st<<"50 12.44";//Поместить в поток

st>>k>>x;//Извлечь из потока

s=x+k;

cout<<s<<endl;//Вывод: 62.44

return 0;

}

**Стандартный класс string**

Класс string расположен в библиотеке **string.lib**.

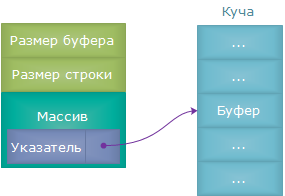
Некоторые преимущества класса string:

1) Класс самостоятельно управляет памятью.

2) Используются перегруженные операции.

3) Отслеживаются ситуации, связанные с выходом за пределы строки.

В памяти строка находится следующим образом:



В памяти объект string хранит размер буфера (количество выделенной в данный момент памяти) и текущий размер строки. Сама строка хранится либо в собственном массиве, либо в выделенной динамически области памяти, указатель на которую хранится вместо массива.

Конструкторы:

string();

string(char\* строка);

Пример. Использование стандартного класса для работы со строками.

#include <iostream>

#include <string>

using namespace std;

int main()

{

string st1; st1="Stroka 1";

cout<<st1<<endl;//Stroka 1

string st2; st2="Stroka 2";

cout<<st2<<endl;//Stroka 2

string st3("Stroka 3");

cout<<st3<<endl;//Stroka 3

st1+="&"+st2+"&"+st3;

cout<<st1<<endl;//Stroka 1&Stroka 2&Stroka 3

return 0;

}

Допустимо использование перегруженной операций сравнения. Для ввода строк обычно используется метод getline.

Некоторые **методы класса string** для работы со строками:

1) const char\* c\_str() - возвращает указатель на нуль-терминальную строку, аналогичную встроенной строке.

2) const char\* data() - возвращает указатель на первый символ строки.

3) size\_t copy(char\* подстрока, size\_t количество\_символов, size\_t позиция=0) - копирует из подстроки в нуль-терминальную строку указанное количество символов начиная с указанной позиции; возвращает количество скопированных символов.

Пример.

string s="ABCDEFGH";

char s1[20];

size\_t slen=s.copy(s1, 2, 3);

s1[slen]='\0';//s1=DE

4) size\_t find(const string &подстрока, size\_t позиция=0) - возвращает номер позиции начиная с указанной, в которой находится подстрока, передаваемая в качестве аргумента; если строка не найдена, функция возвращает -1.

Пример.

string st="ABCDCFCD";

int k=st.find("CD");//k=2

k=st.find("w");//k=-1

k=st.find("C", 3);//k=4

k=st.find("C", 5);//k=6

Имеются перегруженные методы.

5) string substr(size\_t позиция=0, size\_t длина=позиция\_конца\_строки) - возвращает подстроку указанной длины, полученную из строки начиная с указанной позиции.

Пример.

string s="ABCDEFGH";

string s1=s.substr(2, 3);//s1=CDE

6) string& append(const string& строка2) - добавляет строку 2 в конец строки.

Имеются перегруженные методы.

Не обязательно: 7) void push\_back(char символ) - добавляет символ в конец строки.

8) string& insert(size\_t позиция, const string& строка2) - помещает строку 2 в указанную позицию исходной строки.

Имеются перегруженные методы.

Пример.

string s="ABCDСFСD";

s.insert(3, "www");//s=ABCwwwDCD

9) string& erase(size\_t позиция=0, size\_t количество\_символов=размер\_строки) - стирает из строки указанное количество символов начиная с указанной позиции.

Пример.

string s="ABCDСFСD";

s.erase(4, 2);//s=ABCDCD

s.erase();//Удалится вся строка

10) void pop\_back() - удаление последнего символа строки.

11) size\_t size() - возвращает длину строки.

12) size\_t max\_size() - возвращает максимально возможную длину строки.