Глава 8. **Библиотека классов ввода-вывода C++**

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

8.1 Стандартные консольные потоки

Классы потоков – реализация соответствующих шаблонов.

Классы консольных потоков:

`istream – для ввода;`
`ostream – для вывода;`
`iostream – для ввода-вывода.`

Стандартные потоки, связанные с консолью:

`cin –` стандартный ввод (обычно с клавиатуры);
`cout –` стандартный вывод (обычно на экран);
`cerr –` вывод сообщений об ошибках на экран;
`clog –` буферированный вывод сообщений об ошибках на экран.

8.2 Операции «извлечение» и «вставка»

Операции сдвигов "<<", ">>" в классах потоков переопределены для обозначения операций ввода-вывода с преобразованием к символьному виду или из него:

```
ostream& ostream::operator<<(Тип Op)
istream& istream::operator>>(Тип Op)
```

где типы: **char, signed и unsigned short, int, long, float, double, long double, char *(строка), void *(адрес)**

Первый параметр – поток, второй – данные указанных типов, результат – ссылка на тот же поток, что позволяет строить выражения для ввода и вывода.

Примеры записи операций извлечения и вставки

а) `cout << "Input integer:";` // вставка в поток строки

б)
```
int a;
  cin >> a;                          // извлечение из потока
```

в)
```
cout << 2+a+8;
        =>    cout<<(2+a+b);
```

г)
```
cout << "String:" << str << endl;
        =>    ((cout<<"String:")<<str)<<endl;
```

д) `cout << (a<<2);`

е) `cout << (a=2);`

8.3 Управление потоками ввода-вывода Форматирование ввода-вывода. Флаги

```
enum
{ skipws = 0x0001,// пропустить пробелы при вводе
   left   = 0x0002,// выполнять по левой гр. при выводе
   rigth  = 0x0004,// выполнять по правой гр. при выводе
   interval=0x0008,// дополнить пробелами при выводе
   dec    = 0x0010,// преобразовать в десятичную с/с
   oct    = 0x0020,// преобразовать в восьмиричную с/с
   hex    = 0x0040,// преобразовать в шестнадцатир. с/с
   showbase=0x0080,// показывать основание с/с при выводе
   showpoint=0x0100,// показывать дес. точку при выводе
   uppercase=0x0200,// вывод шестн. цифр в верхнем р-ре
   showpos = 0x0400,// выводить + перед полож. числами
   sientific=0x0800,// вывод в формате с плав. точкой
   fixed   = 0x1000,// вывод в формате с фикс. точкой
   unitbuf = 0x2000,// стереть все потоки после вставки
   stdio   = 0x4000};//стереть после вставки stdin,stdout
```

Флаги (2)

Флаги вместе с другими управляющими полями объявлены в классе ios:

```
class ios
{private:
 long x_flags;      // флаги
 int x_width;       // ширина поля вывода
 int x_precision;   // число цифр дробной части
 int x_fill; …}     // символ-заполнитель при выводе
```

Для работы с полями используют специальные методы:

```
flags(), precision(), setf(), width()
```

Пример:

```
cout.setf(ios::uppercase);
cout.setf(ios::hex,ios::dec&oct&hex);
```

8.4 Манипуляторы

Манипулятор – метод, упрощающий настройку потока.

1) Манипуляторы без параметров (объявлены в **<iostream>**):

`ostream & <Имя> (ostream & Os)`

`dec` – десятичная система счисления;

`hex` – шестнадцатеричная система счисления;

`oct` – восьмеричная система счисления;

`ws` – удаление из входного потока пробелов и знаков табуляции;

`endl` – добавление маркера «конец строки» + вывод из буфера;

`flush` – вывод из буфера.

2) Манипуляторы с параметрами (объявлены в **<iomanip>)**:

`setbase(int n)` – установка системы счисления (0, 8, 10, 16);

`setprecision(int n)` – определяет количество дробных цифр;

`setw(int n)` – определяет минимальную ширину поля вывода.

Пример:

```
cout << setw(6) << setprecision(2) << b;
```

8.5 Переопределение извлечения и вставки для объектов пользовательских классов

Переопределяются как функции-операции (т.е. вне класса) :

```
ostream & operator<<(ostream &out,<Тип> <Имя>)
{ ...
  out <<...
  return out;
}

istream & operator>>(istream &in,<Тип> &<Имя>)
{ ...
  in >>...
  return in;
}
```

Пример переопределения операций извлечения и вставки (Ex8_01)

```
#include <iostream>
#include <iomanip>
using namespace std;
class TVector
{
 private: int x,y,z;
 public:  TVector() {}
    friend ostream& operator<<(ostream &stream,
                               TVector obj);
    friend istream& operator>>(istream &stream,
                               TVector& obj);
};
```

Пример переопределения извлечения и вставки

```
ostream& operator<<(ostream &stream,TVector obj)
{
    stream << "Value :";
    stream << setw(5)<< obj.x<<", "<< obj.y;
    stream << ", " << obj.z << "\n";
    return stream;
}
istream& operator>>(istream &stream, TVector& obj)
{
    cout<<"Input value:";
    return stream >> obj.x >> obj.y >> obj.z;
}
```

Тестирующая программа

```
int main(int argc, char* argv[])
{
    TVector A,B;
    cin>>A>>B;
    cout<<A<<B;
    return 0;
}
```

8.6. Создание и настройка объектов-потоков

Классы потоков, связанные с файлами:

ifstream – для ввода из файла;

ofstream – для вывода в файл;

fstream – может использоваться для создания потоков для ввода и вывода.

Объявление потока и открытие файла с помощью конструктора:

```
ifstream in("d:\\Test1.txt"); // для ввода из файла
ofstream out("d:\\Test2.txt"); // для вывода в файл, если файл существовал, то его содержимое стирается
fstream out("d:\\Test2.txt"); // для ввода из файла и вывода в файл
```

При объявлении потока без указания параметров файл не открывается.

Открытие файла

```
void open (const char* filename,int mode);
```

где **mode** - режим ввода/вывода:

in - открыть поток для ввода;

out - открыть поток для вывода;

ate - установить указатель потока на конец файла (отсчет позиции с конца),

app - открыть поток для добавления,

trunc - удалить содержимое файла, если он уже существует,

binary - открыть в двоичном режиме.

Примеры:

```
fstream f; // объявление переменной-потока без открытия файла
```

а) `f.open("simple.txt", ios::in);` // открыть поток для ввода

б) `f.open("simple.txt", ios::out|ios::trunc);` /* открыть поток для вывода и стереть файл с указанным именем, если он существует */

в)
```
f.open("simple.txt",
    ios::in|ios::out|ios::binary); /* открыть двоичный файл для ввода и вывода */
```

Закрытие файла

```
void close();
```

Пример :

```
fstream f;

f.open("simple.txt",ios:: in);

 <Обработка компонентов файла>

f.close();
```

Контроль ошибок при выполнении операций

`int good();` - возвращает ненулевое значение, если при выполнении потоковой операции не возникает ошибки;

`int fail();` – возвращает ненулевое значение, если при выполнении потоковой операции возникает ошибка;

`!` - перегруженная операция - применяется к экземпляру потока для определения состояния ошибки.

Пример:

```
if(!in)          // если файл не открыт
{<выдать сообщение об ошибке>}
else …
```

Пример записи и чтения из файла (Ex8_03)

```
#include <fstream>
#include <iostream>
using namespace std;
void main()
{
   char h;
   ofstream out("Test.txt",ios::app); // для добавления
   cin >> h;
   out << h << endl; // добавляем символ
   out.close();
```

Пример записи и чтения из файла (2)

```
  ifstream  in("Test.txt");   // для ввода из файла
  if(!in)          // если файл не открыт
      cout << "File is not open.\n";
  else
  {
      in >> h;
      while (!in.eof())
      {
            cout << h << endl;
            in >> h;
      }
  }
  in.close();
}
```

8.7 Обработка текстовых файлов

Для выполнения ввода из текстовых файлов и вывода в них используют:

- << - операцию вставки в поток строк, символов или чисел с преобразованием из внутреннего в символьное представление;
- >> - операцию извлечения из потока строк, символов и чисел с преобразованием из символьного во внутреннее представление.

Кроме этого для работы с символами и строками используют:

- перегруженную функцию-метод чтения символа или строки до ограничителя из потока **`get();`**
- функцию-метод чтения строки из потока включая ограничитель **`getline()`**;
- функцию-метод вывода символа в поток **`put()`**.

Ввод символа get()

```
int get(); // возвращает код символа или EOF
```

Пример:

```
#include <iostream>
using namespace std;

int main()
{
   char ch;
   cout << "Enter character";
   while(ch = cin.get())
        if (ch == '\n')break; else cout << ch;
   return 0;
}
```

Ввод символа get()

```
istream &get(char &ch); // вводит символ и возвращает поток
```

Пример:

```
#include <iostream>
using namespace std;

int main()
{
   char ch1,ch2;
   cout << "Enter charactor";
   cin.get(ch1) >> ch2;
   cout << ch1 << ' ' << ch2 << endl;
   return 0;
}
```

Ввод символа get()

/* вводит строку до ограничителя или указанной длины (ограничитель не вводится) */

```
istream& get(char*, int len, char delim= '\n');
```

Пример:

```
#include <iostream>
using namespace std;

int main()
{
   char st[20],ch2;
   cout << "Enter string: ";
   cin.get(st,20,'#') >> ch2;
   cout << st << ' ' << ch2 << endl;
   return 0;
}
```

Ввод строки getline()

```
istream& getline(char* buffer,int size,
                 char delimiter='\n');  /* вводит строку до
                                           ограничителя и ограничитель */
```

Параметры:

buffer - указатель на строку, принимающую символы из потока;

size - задает максимальное число символов для чтения;

delimiter - указывает разделяющий символ, который вызывает прекращение ввода строки до того, как будет введено количество символов, указанное в параметре size (по умолчанию = '\n').

Пример работы с текстовым файлом (Ex8_4)

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   int n = 50;

   ofstream out;  // создаем объект потока

   // открываем файл для вывода
   out.open("Test.txt",ios::out);

   out << "Hello!\n" << n;  // выводим в файл строку и число

   out.close();  // закрываем файл
```

Hello!
50

В шестнадцатеричном виде:
48 65 6C 6C 6F 21 0D 0A 35 30 1A

Пример работы с текстовым файлом (2)

```
  ifstream in("Test.txt"); // открываем файл для ввода
  if (!in)
  {
  cout << "Error.\n";
  }
  else
  {
  char str[80];
  in >> str >> n; // вводим строку и число из файла
  cout << str << " " << n << endl;
  in.close(); // закрываем файл
  }
  return 0;
}
```

Пример использования функции getline() (Ex8_5)

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
   char textLine[50];
   ifstream f("text.txt");
   if (!f) cout<< "Error"<<endl;
   else {
       while (f.getline(textLine,50))
              cout << textLine << endl;
       f.close();
   }
   return 0;
}
```

Если:
а) размер строки превышает заданное ограничение,
б) программа доходит до конца файла,
то getline() возвращает false!

8.8 Обработка двоичных файлов

Для работы с двоичными файлами перегружены функции **`read()`** и **`write()`**:

```
ostream& write(const char* buffer, int num);
ostream& read(char* buffer, int num);
```

`buffer` - это указатель на буфер, содержащий данные;

`num` - указывает число байт, передаваемых в поток или извлекаемых из него.

Пример использования write() (Ex8_6)

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   const int MAX = 80;
   char buff[MAX+1] = "Hello World!";
   int len = strlen (buff) + 1;
   fstream f;
   f.open("CALC.DAT", ios::out|ios::binary);
   f.write((const char*) &len, sizeof(len));
   f.write((const char*) buff, len);
   f.close();
   return 0;
}
```

0D 00 00 00 48 65 6C 6C 6F 20 57 6F 72 6C 64 21 00

Пример использования read() (Ex08_07)

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   const int MAX = 80;
   char buff [MAX+1];
   int len;
   fstream f;
   f.open("CALC.DAT", ios::in|ios::binary);
   f.read((char *) &len, sizeof(len));
   f.read((char *) buff, len);
   cout << len << ' ' << buff << endl;
   f.close();
   return 0;
}
```

13 Hello World!

Файловый ввод/вывод с прямым доступом

Для перемещения файлового указателя используют функцию-метод seekg():

```
istream&  seekg(long pos);
istream&  seekg(long offset, seek_dir dir);
```

pos - номер байта в потоке;

offset - относительное смещение в зависимости от аргумента **dir**:

ios::beg – с начала файла;

ios::cur – с текущей позиции файла;

ios::end – с конца файла.

Пример прямого доступа (Ex08_08)

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   char bbb[80],buff[80] = "Hello World!";
   fstream f,f1;
   f.open("CALC.DAT", ios::out | ios::binary);
   f.write(buff,strlen(buff));
   f.close();
   f1.open("CALC.DAT", ios::in|ios::binary);
   f1.seekg(3); // продвинуться к байту 3
   f1.read(bbb, 5);
   cout << bbb << endl;
   f1.close();
   return 0;
}
```