6. Библиотека fstream - работа с файлами

fstream

ifstream - потоковый ввод из файла, данные только последовательно читаются, для чтения можно использовать операцию **>>**;

ofstream - потоковый вывод в файл, данные только последовательно записываются, для вывода можно использовать операцию **<<**;

fstream - потоковый ввод-вывод в файл, данные и читаются и записываются в конец файла, используются операции **>>**, **<<** соответственно

В каждом из этих классов имеются **конструкторы** с аргументом **char *** для того, чтобы связать поток с конкретным файлом и разрешить работу с ним.

Также связать файл с потоком можно виртуальной функцией **open** – открыть файл.

Функция **close()** закрывает файлы: формирует признак конца файла EOF (End Of File) для выходных файлов, и запрещает потоку работать с ними.

Примеры работы с файлами

Пример 1. Пусть имеется файл целых чисел с именем “f1.dat”. Читая числа из файла, положительные перепишем в файл с именем “f2.dat”.

```
#include <fstream.h>
void main()
{ ifstream f(“f1.dat”); /* конструктор с аргументом char *
    связывает файл с  именем f1.dat с потоком ввода f */
    // или можно так: ifstream f;  f.open(“f1.dat”);
  ofstream g(“f2.dat”); /* конструктор с аргументом char *
          связывает  поток вывода g с файлом “f2.dat”.
    Если такой файл уже был, то он будет
    разрушен. */
```

```
int    x;
while  (f >> x)
    /* прочитать число из файла “f1.dat” и
    записать его в x. Если все числа прочитаны,
    т.е. файл закончился, то операция
    возвращает 0 (NULL) */
 if  (x > 0)    g << x << ’  ‘;
    // записать число x > 0 в файл “f2.dat”
 f.close(); g.close(); /* файлы закрыты, больше
                          их использовать нельзя!*/
}
```

eof()

Проверку на конец файла можно делать и специальной функцией-предикатом **eof()**. Она возвращает *истину*, если файл *закончился*, и *ложь*, если *нет*.

while (!f.eof()) {...}

По умолчанию файлы считаются **текстовыми**, т.е. если это числа, то они преобразуются в символьный вид при выводе и из символьного в числовой при вводе.

Атрибуты ios

Конструкторы этих классов, а также функция **open,** могут иметь дополнительный **второй** аргумент, конкретизирующий способ работы с файлом (**атрибут**).

Атрибут задается как *перечислимый тип* (enum), определенный в базовом классе **ios**.

ios::in – открыть файл для ввода
(для объектов **ifstream** - по умолчанию),

ios::out – открыть файл для вывода
(для объектов **ofstream** - по умолчанию),

ios::app – открыть файл для записи в конец файла, если файла еще нет – создается,

ios::binary – открыть файл как двоичный,
то есть не выполняется преобразование в символьную форму (удобен для чтения-записи структур).

Операция |

Если для файла надо задать **несколько атрибутов**, то они связываются операцией логическое **«или» (|)**, которая трактуется как **объединение** атрибутов.

```
ofstream f(“ff.dat”, ios::app | ios::binary);
/* открыть двоичный файл для
   дополнения */
```

Пример 2.

```
#include <iostream.h>
#include <fstream.h>
void main()
{ int   i;
  ofstream   f; f.open("f4.dat");
                  // или ofstream f("f4.dat");
                 // для дополнения ios::app);
    for (i = 0; i < 20; i++)
     f << rand() % 20 << '  ';
   f.close();
```

Статический массив файлов

```
ofstream  g[3] = {ofstream("g1.dat“),
  ofstream("g2.dat“), ofstream( "g3.dat“)};
            // массив для 3-х файлов вывода
 g[0] << 555 << '  ';
 g[1] << "666 + 777";
 g[2] << 45.78 << ‘  ‘;
 for(i = 0; i < 3; i++)
            g[i].close(); // закрыли
```

Динамический массив файлов

Так как работает только конструктор по умолчанию

```
ofstream  *t = new ofstream [2];
  // массив из 2-х потоковых файлов вывода,
  // но связи с реальными файлами нет
 t[0].open("a.txt");  // связать с
 t[1].open(“b.txt"); // конкретными файлами
 t[0]<<“Маша "; //  запись строк
 t[1]<<“ела"; //  в файлы
 t[0].close(); t[1].close(); // закрыли файлы
```

```
ifstream r("a.txt"); // открыли поток r для чтения
  char s[15];
  f.open("plus.txt"); /* открыли для записи
   выходной поток f, используется повторно */
  r>>s; f<<s; /* чтение строки до ' ' из потока
                 r и запись в поток f (Маша)*/
  cout<<s;
  r.close();  // закрыли работу с файлом «a.txt»
```

Только для чтения!

r.open(“b.txt"); /* открыли повторно
входной поток **r** для чтения
из файла «b.txt» */

r**>>**s; f<<‘ ‘<<s; /* прочитали строку из
файла «b.txt»(ела) и записали
ее в файл «plus.txt»(Маша ела) */

f.close(); // закрыли оба

r.close(); // файла

выше было описание
ifstream r("a.txt");
ofstream f;

Динамический объект - файл

```
ifstream *ff = new ifstream("plus.txt");
                    // открыли поток ввода
                    // в динамической памяти для
  // чтения из файла “plus.txt”
(*ff)>>s;    // прочитали из него строку до ’ ‘
 cout<<"s = "<<s; // вывели на экран
 ff->close(); // закрыли
```

s = Маша

Поток ввода-вывода

```
fstream z("plus.txt", ios::app | ios::out);
//открыли поток вывода для дополнения
 z<<“ кашу!";
        // записали в конец его новую строку
  z.close();     // закрыли файл “plus.txt”
```

```
z.open("plus.txt", ios::in);
   // и открыли этот же поток z только
   // для  чтения
z.getline(s,80);  // прочитали всю строку
cout<<endl<<s; // и вывели на экран
z.close();          //  закрыли
}
```

Маша ела кашу!