Файлы и потоки

Файл – одна из самых общих структур данных в информатике; это «мост» к приложениям и архитектуре ЦВМ. Можно дать определение файла как неограниченной структуры линейно упорядоченных данных, где элементы одинаковой природы могут повторяться. Любой файл можно «открыть» или «закрыть» для информационного обмена с другими частями программы. Можно или читать информацию из файла (входной файл), или писать её в файл (выходной файл), или выполнять обе операции. В первых двух случаях только следующая позиция доступна для чтения или записи соответственно; третий случай обеспечивает доступ к двум позициям.

  


           Текущая позиция

Из-за неограниченности размера файлы обычно хранятся на устройствах внешней памяти, например, магнитных дисках и лентах. В программе каждый файл имеет собственное имя (как и любая другая структура данных или переменная) и своё имя на диске.

В C++ файлы обмениваются информацией с потоками данных; можно сказать, что файлы имеют тип потоков данных. Эти потоки реализованы в классах **ifstream, ofstream** и **fstream**. Итак, можно объявить потоки файлов как

**#include <fstream>**

**ifstream in;**

**ofstream out;**

**fstream general;**

После этого файлу можно дать имя и открыть его для обмена информацией. Это можно сделать двояко:

**string  myFileName;**

**cin >> myFileName;**

**ifstream inFirst(myFileName.data());**

или

**out.open(“veryNewFile.dat”)**

При использовании **fstream** надо указать, для чего открыт файл: для ввода (**ios::in**), вывода (**ios::out**) или для пополнения (**ios::app**); **ios::in** и **ios::out** стандартны для объектов **ifstream** и **ofstream** соответственно.

**fstream generalFirst(“/myFolder/mySub/newFile.dat”,ios::app)**

После попытки открытия обычно проверяют её успешность

**if (out.is\_open())**

**{**

**// we can work with file.**

**}**

Стандартно можно выводить данные следующих типов: **char, short, int, long, char \*, float, double, long double** и **\*void**.

Можно переопределить оператор **<<** для вывода данных других типов.

**class point {float x, y;}**

**ostream & operator << (ostream & s, point & p )**

**{ s << p.x << “  “ << p.y << “  “; }**

**point p(1.0, 2.3);**

**cout << p;**

Аналогично можно поступить с оператором **>>**.

В объектах файловых потоков есть дополнительные функции-члены для обмена с файлами.

Функция **getline** читает строку данных, включая пробельные символы

**char str[60];**

**DataFile.getline(str, 61,’$’);**

**str** – имя символьного массива или указатель на область памяти.

**61** – величина, на единицу большая максимального количества читаемых символов.

‘**$’** – любой символ – ограничитель. При обнаружении этого символа ввод немедленно прекращается. При отсутствии этого параметра по умолчанию используется ‘\n’.

Функция **get** читает один символ из файла, а **put** – пишет один символ в файл.

**inFile.get(ch);**

**outFile.put(ch);**

Предыдущие примеры работали с текстовыми файлами, но наиболее общий случай – простые **двоичные** данные. При этом используется описатель **ios::binary** (группу описателей можно объединить оператором **|**)

**File.open(“inf.dat”, ios::out | ios::binary);**

Для записи в файл **двоичных** данных используют функцию-член файлового потока **write.** При этом в файл можно записать целые массивы и структуры

**File.write((char \*) buffer, sizeof(buffer));**

Всё предыдущее касалось последовательного доступа к файлам, что замедляет работу программ. Вот почему в потоковые объекты включены функции-члены **seekp** и **seekg**, например:

File.seekp(32L, ios::beg);

File.seekp(-10L, ios::end);

File.seekp(120L, ios::cur);

File.seekg(2L, ios::beg);

File.seekg(-100L, ios::end);

File.seekg(40L, ios::cur);

File.seekg(0L, ios::end);

Pos= outFile.tellp();

Pos1=inFile.tellg();

Пусть есть текстовый файл. Напишем программу для вычисления размера этого файла (в байтах) и записи этого размера в конец файла. При обнаружении символа конца строки в конец файла записывается его текущий размер. Например, для текстового файла

**Abcd**

**Efg**

**Hi**

**J**

должен получиться новый файл

**Abcd**

**Efg**

**Hi**

**J**

**5 9 12 14 24**

**#include <iostream>**

**#include <fstream>**

**void main()**

**{**

**fstream inOut(“copy.out”, ios::in | ios::app);**

**int cnt = 0;**

**char ch;**

**while(inOut.get(ch))**

**{**

**cout.put(ch);**

**cnt++;**

**if (ch == ‘\n’)**

**inOut << cnt << ” “;**

**}**

**inOut << cnt << endl;**

**cout << “[ “ << cnt << “ ]” << endl;**

**}**

**Result 1**

**[ 0 ]**

**#include <iostream>**

**#include <fstream>**

**void main()**

**{**

**fstream inOut(“copy.out”, ios::in | ios::app);**

**int cnt = 0;**

**char ch;**

**inOut.seekg(0, ios::beg);**

**while(inOut.get(ch))**

**{**

**cout.put(ch);**

**cnt++;**

**if (ch == ‘\n’)**

**{**

**inOut << cnt << “ “;**

**}**

**}**

**inOut << cnt << endl;**

**cout << “[ “ << cnt << “ ]” << endl;**

**}**

**Result 2**

**Abcd**

**[ 5 ]**

**#include <iostream>**

**#include <fstream>**

**void main()**

**{**

**fstream inOut(“copy.out”, ios::in | ios::app);**

**int cnt = 0;**

**char ch;**

**inOut.seekg(0, ios::beg);**

**while(inOut.get(ch))**

**{**

**cout.put(ch);**

**cnt++;**

**if (ch == ‘\n’)**

**{**

**streampos mark = inOut.tellg();**

**inOut << cnt << “ ”;**

**inout.seek(mark);**

**}**

**}**

**inOut << cnt << endl;**

**cout << “[ “ << cnt << “ ]” << endl;**

**}**

**Result 3**

**Abcd**

**Efg**

**Hi**

**J**

**[ 24 ]**

**#include <iostream>**

**#include <fstream>**

**void main()**

**{**

**fstream inOut(“copy.out”, ios::in | ios::app);**

**int cnt = 0;**

**char ch;**

**inOut.seekg(0, ios::beg);**

**while(inOut.get(ch))**

**{**

**cout.put(ch);**

**cnt++;**

**if (ch == ‘\n’)**

**{**

**streampos mark = inOut.tellg();**

**inOut << cnt << “ ”;**

**inout.seek(mark);**

**}**

**}**

**inOut.clear();**

**inOut << cnt << endl;**

**cout << “[ “ << cnt << “ ]” << endl;**

**}**

**Result 5**

**Abcd**

**Efg**

**Hi**

**J**

**[ 24 ]**