Лекция 13-2

Ввод и вывод

Работа с файлами

Файлом называется поименованная часть памяти диска, содержащая некоторый набор записей. Непосредственную работу с файлами выполняет операционная система. В языках программирования имеются высокоуровневые средства для доступа к файлам. В C++ они объявлены в заголовочном файле **fstream**.

Источник или приемник данных называется ***потоком***. Примерами потоков являются стандартные потоки для ввода cin и для вывода cout.

Для чтения из файла или записи в файл создаются потоки, связанные с файлами. Существуют два вида потоков: ***текстовые*** и ***бинарные***.

Текстовый поток – это последовательность символьных строк, заканчивающихся символом «новая строка» - '\n'. При выводе перед каждым символом '\n' помещается символ возврата каретки '\r'. При вводе происходит обратное преобразование – два символа '\r' и '\n' заменяются одним символом '\n'. Необходимость такого преобразования вызвана различиями в фиксации признака новой строки в C++ и в операционной системе. В языке C++ новая строка обозначается единственным символом '\n', реально же на диске в файле хранятся оба символа '\r' и '\n', генерируемые нажатием клавиши **Enter**, которая используется при завершении ввода очередной строки с помощью клавиатуры.

Бинарный поток – это последовательность байтов, которая не

Стандартные потоки

При каждом пуске программы открыты три ***стандартных потока***:

cin – стандартный ввод,

cout – стандартный вывод,

cerr – стандартный приемник ошибок.

По умолчанию стандартный ввод связан с клавиатурой, стандартный вывод – с экраном дисплея.

Стандартный ввод, и вывод можно перенаправить на файлы средствами операционной системы. Эти потоки имеют буфера, в которые предварительно направляются данные. Это позволяет ускорить ввод и вывод.

Поток cerr не имеет буфера, его нельзя перенаправить на файл. Этот поток всегда связан со стандартным устройством вывода, обычно это экран монитора.

Файловые потоки

В заголовочном файле fstream объявлены три потоковых класса, обеспечивающие работу с файлами:

ofstream – класс выходных файловых потоков;

ifstream – класс входных файловых потоков;

fstream – класс двунаправленных файловых потоков.

Для работы с файлами в программе нужно определить переменные этих классов, например,

```
ifstream fin;        // fin - входной файловый поток
ofstream fout;       // fout - выходной файловый поток
```

Связь потоков с файлами

Связь потока с файлом можно установить с помощью функции-члена **open()**, с двумя аргументами. Первый аргумент обязателен, он задает имя открываемого файла. Второй аргумент не является обязательным, он задает режим работы с файлом. Если второй аргумент на задан, файлы считаются текстовыми.

Примеры:

```
fin.open("InpData.txt");       // Файл InpData.txt открывается для чтения
fout.open("ResData.txt");    // Файл ResData.txt открывается для записи
```

В качестве второго аргумента функции open() можно использовать константы (называемые флагами), определенные в классе ios:

ios::out – файл используется для вывода;

ios::in – файл используется для ввода;

ios::app – файл открывается для добавления в конец файла;

ios::binary – файл открывается как бинарный.

Примеры:

```
fin.open("InpData.txt", ios::in); // Файл открывается для чтения
```

Связь потоков с файлами

Для проверки успешности завершения функции open() к потоку можно применить оператор !, который возвращает ненулевое значение при наличии ошибки. Например, проверить, что файл InpData.txt удалось открыть, можно следующим образом:

```
if( !fin != 0 )
  {
      cerr << "Не могу открыть файл " << "InpData.txt";
      exit(1); // Завершение работы программы
  }
```

По умолчанию потоки открываются на чтение и запись как текстовые. Для текстовых файловых потоков применимы все рассмотренные ранее операторы и функции ввода и вывода.

Открытые файловые потоки закрываются автоматически при завершении программы или при выходе из функции, в которой они были созданы. Поток можно явно закрыть, вызвав для него функцию **close()**.

Пример:

```
fin.close();
```

Программа «Копирование файлов»

Открываются два файловых потока - один на чтение, другой на запись. Из входного файла производится построчное чтение, прочитанные строки выводятся на экран и в выходной файл. После чтения каждых 20 строк программа останавливается и продолжает выполняться после нажатия клавиши **Enter**.

```
#include <fstream>
#include <iostream>
#include <cstdlib> // Для exit()
using namespace std;
const int PAGE = 20;
// Максимальное число строк на странице
void main()
{ setlocale(LC_ALL, "Russian");
ifstream fin; // Файловый поток для ввода
ofstream fout; //Файловый поток для вывода
char filename[20]; // Строка для имени файла
char line[100]; // Строка из файла
cout << "Введите имя входного файла ";
cin.getline(filename,20);
// Ввод имени входного файла
fin.open(filename);
if( !fin ){ // Если не удалось открыть входной файл
cerr << "Не удалось открыть файл " << filename;
system("pause");
exit(1);  } // Завершение программы
cout << "Введите имя выходного файла ";
cin.getline(filename,20);
fout.open(filename);
if(!fout){
// Если не удалось открыть выходной файл
cerr << "Не удалось открыть файл " << filename;
system("pause");
exit(1);}
int i = 0; // Количество прочитанных строк
while( !fin.eof() ){ // Пока не достигнут конец файла
fin.getline(line,100); // Чтение строки из файла.
i++; // Увеличение счетчика
cout << line << '\n'; // Вывод строки на экран
fout << line << '\n'; // Вывод строки в файл
if( i % PAGE == 0 ) // Если выведена очередная страница,
cin.get(); // ждем нажатия клавиши Enter
}
fin.close(); // Закрытие входного файла
fout.close(); // Закрытие выходного файла
system("pause");
}
```

Программа «Подсчет суммарных затрат»

В текстовом файле «input.txt» содержатся записи, содержащие в себе фамилию покупателя и стоимость покупки. Каждая запись размещена на отдельной строке. Фамилии в записях могут повторяться. Записать в файл «output.txt» в порядке возрастания затрат фамилии покупателей и их суммарные затраты.

```
#include "fstream"
#include "string.h"
using namespace std;
int main()
{
	ifstream fin("input.txt");
	ofstream fout("output.txt");
	int i,j,p;
	int n=0;// количество покупателей
	char s[30];
	char* name[100];//массив покупателей
	char* c;
	int prise[100];//массив суммарных затрат
	while(!fin.eof())// пока не конец файла
	{
		fin>>s;//считываем фамилию покупателя
		fin>>p;//считываем стоимость покупки
		i=0; // поиск фамилии в списке покупателей
		while(i<n && strcmp(s,name[i])!=0)
			i++;
		if (i<n)// покупатель был
			prise[i]+=p;// увеличиваем его затраты
		else
		{	//добавление новой записи
			name[n]=new char[strlen(s)+1];
			strcpy(name[n],s);
			prise[n]=p;
			n++;
		}
	}
for(i=0;i<n-1;i++)// сортировка
	for(j=i+1;j<n;j++)
		if(prise[i]>prise[j])
		{
			p=prise[i];
			prise[i]=prise[j];
			prise[j]=p;
			c=name[i];
			name[i]=name[j];
			name[j]=c;
		}
for(i=0;i<n;i++)//вывод в файл
	fout<<name[i]<<" "<<prise[i]<<endl;
for(i=0;i<n;i++)// очистка динамической памяти
	delete name[i];
return 0;}
```

Работа с бинарными файлами

Связать файловые переменные с конкретными файлами на диске можно, передавая конструктору имена файлов и режим работы с ними.

Пример:

```
ofstream textout(NameTextFile, ios::out);
```

создает текстовый файловый поток textout, связывает его с файлом, имя которого находится в строке NameTextFile и открывает его на запись.

```
ofstream binout(NameBinFile, ios::out | ios::binary);
```

создает бинарный выходной поток binout (флаги ios::binary и ios::out), связанный с файлом, имя которого указано в строке NameBinFile.

Замечание: данные флаги представляют из себя целые числа, все разряды которых нули, кроме одного разряда, поэтому их можно комбинировать с помощью операции логического или (|).

Для чтения из бинарного потока применяют функцию:

```
int read(char *pc, size_t count);
```

которая читает из потока count байтов и сохраняет их в массиве, начало которого указывает pc.

Для записи в бинарный поток служит функция:

```
int write(const char *pc, size_t count);
```

направляющая в поток count байтов из массива, на который указывает pc.

Работа с бинарными файлами

Позиции в файле нумеруются с нуля, поэтому бинарный поток подобен массиву байтов. С каждым файловым потоком связан текущий указатель на байт, который будет прочитан или записан при следующей операции ввода/вывода.

Положением текущего указателя в ***выходном*** потоке можно управлять с помощью функции:

ostream& seekp(long offs, seek_dir dir);

Текущую позицию во ***входном*** потоке можно изменить функцией

istream& seekg(long offs, seek_dir dir);

Первый параметр offs функций seekp() и seekg() задает число позиций, на которое надо переместить текущий указатель; второй параметр dir назначает точку отсчета, от которой надо произвести смещение. Для указания точки отсчета можно использовать перечислимые константы из файла iostream:

enum seek_dir { beg, cur, end };

- beg – смещаться от начала файла,
- cur – смещаться от текущей позиции,
- end – смещаться от конца файла.

Пример: если файловый поток f открывается на чтение и запись, то

Программа «Сравнение текстового и бинарного файлов»

В программе создаются два выходных файловых потока: текстовый и бинарный. В файлы записываются случайные целые числа в текстовом и бинарном виде. Созданные файлы закрываются и открываются вновь как бинарные, для передачи функции FileSize(), определяющей их размеры. Чтобы прочитать содержимое файла, хранящего текстовое представление чисел, он сначала закрывается как бинарный, а затем открывается на ввод как текстовый. Файл с бинарным представлением чисел подготавливается к чтению перемещением текущего указателя к началу файла. Содержимое файлов выводится на экран.

Программа «Сравнение текстового и бинарного файлов»

```
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <ctime>
using namespace std;
long FileSize(istream& f) // Возвращает размер файла в байтах
{                                 // f – ссылка на файловый поток
    long k; // Размер файла в байтах
    f.seekg(0, ios::end); // Поместить текущий указатель в конец файла
                                  // 0 - величина смещения
                                 // ios::end - позиция, относительно которой делается смещение
    k = f.tellg(); // Функция tellg() возвращает текущую позицию в файле
    return k;
}
```

Программа «Сравнение текстового и бинарного файлов»

```
int main()
{
    setlocale(LC_ALL, "Russian");
    const char* NameTextFile = "TxtFile.txt"; // Имя текстового файла
    const char* NameBinFile = "BinFile.bin"; // Имя бинарного файла
    const int N = 20; // Количество чисел в файле
    const int NL = 12; // Количество чисел на одной строке
    // Создание выходного текстового потока
    ofstream textout(NameTextFile, ios::out);
    if(!textout){ // Проверка
        cout<< "Не могу создать файл" << NameTextFile;
        cin.get(); // Ждем нажатия Enter
        exit(1); }
    // Создание выходного бинарного потока
    ofstream binout(NameBinFile, ios::out | ios::binary);
    if(!binout){ // Проверка
        cout<< "Не могу создать файл" << NameBinFile;
        cin.get();
        exit(1); }
```

Программа «Сравнение текстового и бинарного файлов»

```
int i, nmb;
srand((unsigned)time(0)); // Инициализация датчика случайных чисел
for(i = 0; i < N; i++){
     nmb = rand(); // Генерируем числа
     textout << ' ' << nmb; // и заносим в текстовый
     binout.write((char*)&nmb, sizeof(int)); }    // и бинарный файлы
textout.close(); // Закрываем
binout.close(); // потоки
 // Открываем оба файла на чтение как бинарные
 ifstream tin(NameTextFile, ios::binary); // Бинарный поток
 ifstream bin(NameBinFile, ios::binary);
 cout << "Размер текстового файла " << NameTextFile<< " = " << FileSize(tin);
 cout << "\nРазмер бинарного файла " << NameBinFile<< " = " << FileSize(bin);
 tin.close(); // Закрываем бинарный поток, связанный с текстовым файлом
 tin.open(NameTextFile); // и открываем файл как текстовый поток
 if(!tin){
      cout << "\nНе могу открыть файл " << NameTextFile;
     cin.get();
     exit(1);}
```

Программа «Сравнение текстового и бинарного файлов»

```
    // Вывод чисел из обоих файлов
    cout << "\nЧисла из файла " << NameTextFile << endl;
    i = 0;
    while(!tin.eof()){ // Пока не достигнут конец файла,
       tin >> nmb; // читаем число из файла
       i++; // Счетчик чисел
       cout << nmb << ' '; // выводим на экран
       if(i % NL == 0) // Если строка заполнена,
          cout << endl;  }// переходим на новую строку
    bin.seekg(0, ios::beg); // Перемещаем текущий указатель файла к его началу
    cout << "\nЧисла из файла " << NameBinFile << endl;
    i = 0;
    while(!bin.eof()){ // Пока не достугнут конец файла,
       bin.read((char*)&nmb, sizeof(int)); // Читаем байты числа
       i++;
       cout << nmb << ' '; // Выводим число
       if(i % NL == 0)
          cout << endl;  }
tin.close(); // Закрываем
bin.close(); // потоки
cin.get();   return 0;  }
```