Лекция 11

ФАЙЛОВЫЙ ВВОД И ВЫВОД ДАННЫХ В VISUAL C++

Цель лекции. Изучить особенности использования файлов для ввода, хранения и вывода данных в Visual C++.

Основные вопросы лекции.

1. Файлы и потоки ввода и вывода данных в Visual C++ 2010.
2. Создание, открытие и закрытие файлов в Visual C++ 2010.
3. Запись данных в файл и чтение из файла в Visual C++ 2010.
4. Позиционирование файла в Visual C++ 2010.

1. Файлы и потоки ввода и вывода данных в Visual C++ 2010

При решении большинства задач на любом языке программирования возникает необходимость записывать, хранить и получать информацию, используя файлы.

Файл (file) – это именованный объект, хранящий данные (программа или любая другая информация) на каком-либо носителе (винчестер, флешь-память, CD и др.).

Поток (stream) - это виртуальное логическое устройство, связывающее программу с физическим устройством ввода-вывода (терминалом, дисководом и др.). Поскольку потоки не зависят от физических устройств, то одна и та же функция может записывать информацию на диск или на другое устройство.

Поток связывают с определенным файлом, выполняя операцию **открытия**. Как только файл открыт, можно проводить обмен информацией между ним и программой.

Файл отсоединяется от определенного потока (т.е. разрывается связь между файлом и потоком) с помощью операции **закрытия**. При закрытии файла, открытого с целью вывода, содержимое (если оно есть) связанного с ним потока записывается на внешнее устройство. Этот процесс, который обычно называют дозаписью потока, гарантирует, что никакая информация случайно не останется в буфере диска. Если программа завершает работу нормально, то все файлы закрываются автоматически. В случае аварийного завершения программы, файлы не закрываются.

В языке C++ существует два типа потоков:

- текстовый (**text**);
- двоичный (**binary**).

Текстовый поток – это последовательность символов. В **C++** считается, что текстовый поток организован в виде строк, каждая из которых заканчивается символом новой строки. Среди символов в потоке может быть символ возврата каретки, перехода на новую строку и др.

Двоичный поток – это последовательность байтов, однозначно соответствующих информации, находящейся на внешнем носителе. Причем никакого преобразования символов не происходит.

Доступ к информации в потоках и в файлах неодинаков. Например, из файла на диске можно выбрать 3-ю запись или заменить 6-ю запись. В то же время в файл, связанный с печатью, информация может передаваться только последовательно. Это иллюстрирует самое главное отличие между потоками и файлами.

Потоки, используемые в программах, делятся на:

- входные, из которых читается информация;
- выходные, в которые вводится информация;
- двунаправленные, допускающие как чтение, так и запись.

В соответствии с особенностями «устройства», к которому «присоединен» поток, потоки принято разделять на:

- стандартные;
- консольные;
- строчные;
- файловые.

Стандартные и консольные потоки соответствуют передаче данных от клавиатуры до дисплея.

Если символы потока в совокупности образуют символьный массив в основной памяти, то это **строчный поток.**

Если информация размещается на внешнем носителе данных, то это **файловый поток** или просто **файл**.

До сих пор во всех программах курса выполнялся обмен со стандартными потоками:

cin – стандартный входной поток, связанный с клавиатурой;

cout – стандартный выходной поток, связанный с экраном дисплея.

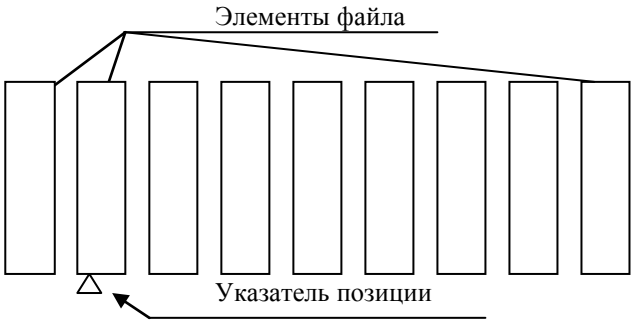
Используя операции включения (записи) данных в поток << и извлечения данных из потока >> выполнялся обмен данными с дисплеем и с клавиатурой ПК. Для этих целей в программу необходимо было включить заголовочный файл **iostream.h**.

Рассмотрим особенности обмена данными с файлами.

2. Создание, открытие и закрытие файлов в Visual C++

Библиотека ввода и вывода данных в файлы подключается в заголовочном файле **stdio.h** и включает средства для работы с последовательными файлами. **Последовательный файл** можно представить как именованную цепочку (ленту, строку) байтов, имеющую начало и конец. Последовательный файл отличается от файла с другой организацией тем свойством, что чтение из файла (или запись в него) ведется байт за байтом от начала до конца.

В каждый момент времени позиция в файле, из которого выполняется чтение (или запись), определяется значениями **указателя позиции записи и чтения файла**.



Установка указателя записи (чтения) на нужные байты выполняется либо автоматически, либо за счет управления их положением. В библиотеке ввода-вывода есть соответствующие средства.

При работе с файлами используются следующие операции:

- создание файла;
- удаление файла;
- поиск файла на внешнем носителе;
- открытие файла;
- чтение из файла или запись данных в файл;
- позиционирование файла;
- закрытие файла.

Все перечисленные действия могут быть выполнены с помощью средств библиотеки ввода-вывода.

Операция **открытия файла** связывает поток с определенным файлом. Операция **закрытия** файла разрывает эту связь.

Запись или чтение из файла осуществляются с помощью указателя файла. **Указатель файла** – это указатель на структуру типа **FILE**. Для объявления переменной–указателя файла, например, ***fp**, используется следующий оператор:

FILE *fp;

Указатель файла указывает на структуру, содержащую различные сведения о файле, его имя, статус и указатель текущей позициив начало файла

При обработке данных, хранящихся в файле, программе необходимо иметь доступ к данному файлу. С помощью переменной ***fp** ведется в дальнейшем вся работа с файлом в программе.

FILE *F1;

В файле **stdio.h** определены функции для работы с файлами. Основные из них приведены в таблице 11.1.

Таблица 11.1. Функции для работы с файлами в Visual C++

Функция	Описание функции
fopen()	Открыть файл
fclose()	Закрыть файл
fseek()	Переместить (установить) указатель позиции файла
feof()	Возвращает значение «истина», если достигнут конец файла
ferror()	Возвращает значение «ложь», если найдена ошибка
fread()	Читает блок данных из потока.
fwrite()	Записывает блок данных в поток
rewind()	Устанавливает указатель позиции файла на начало
remove()	Удаляет файл

При открытии файла функция **fopen()** выполняет два действия:

- открывает файл и связывает его с потоком;
- возвращает указатель, ассоциируемый с этим файлом.

Прототип функции **fopen()** имеет следующий вид:

FILE *fopen (char *filename, char mode);

где **char *filename** – строка, содержащая полное имя файла на диске; **char *mode** – строка, определяющая режим открываемогофайла.

Возможны следующие режимы открытия файла:

“**r**” – открыть файл для чтения;

“**w**” – создать файл для записи;

“**a**” – открыть для добавления в существующий файл;

“**rb**” – открыть двоичный файл для чтения;

“**wt**” – создать текстовый файл для записи;
“**at**” – открыть текстовый файл для добавления;
“**r+t**” – открыть текстовый файл для чтения и записи;
“**w+t**” – создать текстовый файл для чтения и записи;
“**a+t**” – открыть текстовый файл для добавления.
Например, для создания файла для записи данных с именем **C:\\Inform.dat** необходимо записать:

```
FILE *F1;
F1= fopen (“C:\\ Inform.dat”, “w”);
```

При открытии файла всегда необходимо провести проверку открытия файла.
Применим полученные сведения для решения реальной задачи.

Пример 1. Исследовать программу для записи переменной **A** в открываемый файл, чтения значения переменной A из этого файла и вывода значения **A** на печать.

```
#include <stdio.h>                          //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                         //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
        double A, B, S;
        double *pS;                                     //Указатель на буфер обмена
        pS=&S;                                          //Присваивание адреса S указателю pS
        cout<<"Vvedite A"<<endl;
        cin>>A;                                         //Ввод A
        cout<<"A= "<<A<<endl;
        FILE*F1;                                        //Объявление файла для записи
        if((F1=fopen("D:\\Inform.DAT", "w"))==NULL)     //Стандартная процедура создания и
                {                                       //открытия файла для записи данных с одно-
                cout<<"Ne mogu otkrit F1"<<endl;        //временной проверкой результата этих
                exit(1);                                //действий и сообщением в случае ошибки
                }

        S=A;                                            //Запись A в буфер
        fwrite(pS,4,1,F1);                              //Запись из буфера в открытый файл
        fclose(F1);                                     //Обязательное закрытие файла

        fread(pS,4,1,F1);                               //Чтение переменной pS из файла
        B=S;                                            //Запись из буфера в переменную B
        cout<<"B= "<<B;
        getch();
        return 0;
}
```

В данной программе используется метод определения ошибки при открытии создаваемого файла. Неоткрытие файла приравнивается к константе **NULL**, которая определена в библиотеке **stdio.h**. Функция **exit(1)** определена в файле **stdlib.h** и прекращает выполнение программы, а **единицу(!) возвращает** в операционную систему. Перед прекращением программы она закрывает все открытые файлы, освобождает буферы и выводит все необходимые сообщения на экран.

Если файл открывается для записи, то существующий файл удаляется и создается новый.

При открытии файла для чтения, требуется, чтобы он существовал.

В случае открытия файла для чтения и записи существующий файл не уничтожается, однако создается, если он не существовал ранее.

После чтения данных из файла (или записи данных в файл) он должен быть закрыт следующим образом:

```
fclose (F1);
```

Отлаженная программа дает следующий результат:

```
Vvedite A
45
A= 45
B= 45
```

Таким образом, значение переменной было верно записано в специально открытый файл **F1** и прочитано из этого файла.

3. Запись (чтение) данных в файл

Запись данных в поток проводится функцией **fwrite()**, а чтение - функцией **fread()**. Эти функции имеют следующие прототипы:

unsigned fread (void *buf, int bytes, int c, FILE *fptr);

и **unsigned fwrite (void *buf, int bytes, int c, FILE *fptr);**

где **buf** – указатель на буфер памяти, откуда будет происходить обмен с файлом:
bytes – длина каждой единицы записи (чтение) в байтах;
c – количество единиц записи, которое будет прочитано (записано);
fptr – указатель на соответствующий файл.

В Примере 1 уже были применены эти операторы для ввода в файл и вывода из него одной переменной, для чего используется буфер памяти компьютера. Исследуем их применение

Пример 2. В памяти ПК есть массив из 12 произвольных чисел. Записать этот массив в файл **D:\\Inform.dat**. Прочитать этот массив из файла и вывести его на экран дисплея.

Программа записи массива в файл, чтение массива из файла и отображения его на экране дисплея будет иметь следующий вид.

```
#include <stdio.h>                          //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                         //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                             //Максимальный размер массива
int A[N];                                   //Объявление массива A[N]
int i,n;                                    //Параметр цикла i и реальный размер массива n
int S;                                      //Объявление переменной S в качестве буфера обмена
int *pS;                                    //Указатель pS на буфер обмена S
pS=&S;                                      //Присваивание адреса переменной S указателю pS
cout<<"Vvedite razmer n:"<<endl;
cin>>n;                                     //Ввод реального размера массива A[N]
cout<<"Vvedite massiv A:"<<endl;
        for(i=0;i<n;i++)                    //Цикл для ввода массива A[N]
        cin>>A[i];                          //Ввод i-го элемента массива A[N]
cout<<" Massiv A: "<<endl;
        for(i=0;i<n;i++)                    //Цикл для вывода массива A[N]
        cout<<A[i]<<" ";                    //Вывод i-го элемента массива A[N]
FILE*F1;                                    //Объявление переменной–указателя файла
if((F1=fopen("D:\\Inform.dat", "w"))==NULL)      //Открытие файла для записи с условием вывода на
{                                                // экран предупреждения в случае сбоя при открытии
cout<<"Ne mogu otkrit F1"<<endl;
exit(1);
}
        for(i=0;i<n;i++)                    //Цикл для записи в открытый файл массива A[N]
        {
        S=A[i];                             //Запись i-го элемента массива A[N] в буфер S
        fwrite(pS,4,1,F1);                  //Запись i-го элемента массива A[N] из буфера в файл
        }
fclose(F1);                                 //Закрытие файла
int B[N];                                   // Объявление нового массива B[N]
FILE*F2;                                    //Объявление переменной–указателя файла
if((F2=fopen("D:\\Inform.dat","r"))==NULL)       //Открытие файла для чтения с условием вывода на
{                                                //экран предупреждения в случае сбоя при открытии
cout<<"Ne mogu otkrit F2"<<endl;
exit(1);
}
        for(i=0;i<n;i++)                    //Цикл для чтения из файла элементов массива B
        {
        fread(pS,4,1,F2);
        B[i]=S;                             //Запись единицы данных из буфера S в i-й элемент массива B
        }
```

```
fclose(F2);                                  //Закрытие файла
cout<<endl<<" New massiv B:"<<endl;
        for(i=0;i<n;i++)                     //Цикл для вывода массива B[N]
        cout<<B[i]<<" ";                     //Вывод на экран i-го элемента массива  B
cout<<endl;
getch();
return 0;
}
```

Вид экрана после выполнения программы:

```
Uvedite razmer n:
6
Uvedite Mas:
3 6 1 7 9 4
 Massiv A:
3 6 1 7 9 4
 New massiv B:
3 6 1 7 9 4
```

При чтении данных из файла размер файла часто неизвестен. Для определения конца файла служит функция **feof()**. Она имеет следующий прототип:

int feof (FILE *fptr);

Функция возвращает значение «истина», если достигнут конец файла и «ложь» - если нет.

Пример 3. Исследуем программу чтения данных из файла **C:\\Inform.dat** с использованием функции **feof()**. Такая программа имеет следующий вид:

```
#include <stdio.h>                           //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                          //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
float S;                                     //объявление буфера обмена
float *pS;                                   //объявление указателя буфера
float A[9];                                  //объявление массива
int i;
pS=&S;                                       //присвоение адреса буфера указателю буфера
cout<<"Vvedite  massiv A:"<<endl;
for(i=0; i<9; i++) cin>>A[i];                //ввод массива
        FILE *F1;                            //объявление указателя на файл F1
if((F1=fopen("C:\\Inform.dat", "w+b"))==NULL)  //открытие файла C:\\Inform.dat в режиме  записи
 {
        cout<<"Ne mogu otkrit file1!"<<endl;
        exit(1);
 }
for(i=0; i<9; i++)
{
        S=A[i];                              //запись элемента массива в буфер
        fwrite(pS, 4, 1, F1);                //запись одной порции из 4-х байтов в файл
 }
fclose(F1);                                  //закрытие файла F1
int l;
float B[9];                                  //объявление массива B
FILE *F2;                                    //объявление указателя файла F2
if((F2=fopen("C:\\Inform.dat", "r+b"))==NULL)  //открытие файла C:\\Inform.dat в режиме  чтения и записи
{
        cout<<"Ne mogu otkrit file2!"<<endl;
        exit(1);
}
i=0;
while(!feof(F2))                             //чтение данных из файла пока не наступит конец файла
{
        fread(pS, 4, 1, F2);                 //чтение одной порции из 4-х байтов в буфер S
        B[i]=S;                              //запись из буфера S в массив
        i+=1;                                //счет количества прочитанных чисел из файла
```

```
}
l=i-1;                                  //определение количества прочитанных чисел
fclose(F2);                             //закрытие файла F2
cout<<"Massiv B:"<<endl;
for(i=0; i<l; i++) cout<<B[i]<<' ';     //вывод массива на экран
cout<<endl;
}
```

Вид экрана после выполнения программы:

```
Vvedite  massiv A:
1 2 3 4 5 6 7 8 9
Massiv B:
1 2 3 4 5 6 7 8 9
```

4. Позиционирование файла

Функция **rewind ()** устанавливает указатель позиции файла на начало файла. Прототип этой функции имеет следующий вид:

```
void rewind (FILE *fptr);
```

Обращение к функции имеет вид:

```
rewind (F2);
```

Чтение и запись в файл не обязательно делать последовательно. Можно получить доступ непосредственно к нужному байту. Для этих целей используется функция **fseek()**, устанавливающая указатель позиции в нужное место.

Прототип этой функции имеет вид:

```
int fseek (FILE *fptr, long num, int mode);
```

где: **fptr** – указатель на соответствующий файл;
num – количество байт от точки отсчета для установки текущей позиции указателя файла;
mode – определяет точку отсчета.

Точка отсчета	Значение **mode**
1. Начало файла	0
2. Текущая позиция	1
3. Конец файла	2

Пример 4. В файле **C:\\Inform.dat** записано 9 чисел: 1 2 3 4 5 6 7 8 9. Необходимо прочитать данные из этого файла, начиная с 3-го числа (то есть с 3), определить сумму прочитанных чисел и добавить ее в файл **C:\\Inform.dat**. Затем прочитать полученный массив из файла и вывести его на экран дисплея.

Программа будет иметь вид:

```
#include <stdio.h>                      //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                     //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int n=30;                         //максимальний размер массива
float S *pS;                            //объявление буфера обмена и указателя на буфер
float Mas1[n], Sum;                     //объявление массива и суммы
int i,l;
pS=&S;                                  //присваивание адреса буфера указателю буфера
FILE *F2;                               //объявление указателя файла F2
if((F2=fopen("C:\\Inform.dat", "r+b"))==NULL)   //открытие файла C:\\Inform.dat в режиме чтения и записи
 {
        cout<<"Ne mogu otkrit file!"<<endl;
        exit(1);
 }
i=0;
fseek(F2, 8, 0);                        //установка указателя позиции файла через 8 байтов с начала файла
while(!feof(F2))                        //чтение данных из файла, пока не наступит конец файла
{
        fread(pS,  4, 1, F2);           //чтение одной порции из 4-х байтов в буфер S
        Mas1[i]=S;                      //запись из буфера S в массив
        i+=1;                           //счет количества прочитанных чисел
```

```
}
l=i-1;                                          //определение количества прочитанных чисел
fclose(F2);                                     // закрытие файла
cout<<endl;
for(i=0; i<l; i++) cout<<Mas1[i]<<'  ';         //вывод массива на экран
cout<<endl;
Sum=0;
for(i=0; i<l; i++) Sum=Sum+Mas1[i];             //вычисление суммы
cout<<"Sum= "<<Sum<<endl;                       //отображение суммы на экране
FILE *F3;                                       //объявление указателя файла  F3
F3=fopen("C:\\Inform.dat", "a");                //открытие файла C:\\Inform.dat в режиме добавления
S=Sum;                                          //пересылаем сумму в буфер
fwrite(pS, 4, 1, F3);                           //запись суммы из буфера в файл
fclose(F3);                                     //закрытие файла F3
FILE *F4;                                       //объявление указателя файла  F4
if((F4=fopen("C:\ \Inform.dat", "r+b"))==NULL)  //открытие файла C:\\Inform.dat в режиме чтения и записи
{
cout<<" Ne mogu otkrit file 2!"<<endl;
        exit(1);
}
i=0;
while(!feof(F4))                                //чтение данных из файла, пока не наступит конец файла
{
        fread(pS,  4, 1, F4);                   //чтение одной порции из 4-х байтов в буфер S
        Mas1[i]=S;                              //запись из буфера S в массив
        i+=1;                                   //счет количества прочитанных чисел из файла
}
l=i-1;                                          //счет количества прочитанных чисел из файла
fclose(F4);                                     //закрытие файла F4
cout<<endl;
for(i=0; i<l; i++) cout<<Mas1[i]<<'  ';         //вывод массива на экран
cout<<endl;
}
```

Вид экрана после выполнения программы:

```
3 4 5 6 7 8 9
Sum= 42
1 2 3 4 5 6 7 8 9 42
```

Выводы. При усложнении программ возникает необходимость хранить и получать информацию, используя файлы.

Операции ввода-вывода в языке С++ организованы с помощью библиотечных функций.

При работе с файлами используются следующие процедуры:

- создание файлов;
- удаление файлов;
- поиск файлов на внешнем носителе;
- открытие файла;
- чтение из файла или запись данных в файл;
- позиционирование файла;
- закрытие файла.

Все перечисленные действия могут быть выполнены с помощью средств библиотеки ввода-вывода.

Вопросы для самоконтроля.

1. Для ввода данных в файл необходимо следующее объявление:
   1. **ofstream name_file**;
   2. **ofstream (“filename.txt”)**;
   3. **ofstream name_file (“filename.txt”)**;
2. Чтобы выполнить операцию вывода данных из файла необходимо следующее объявление:
   1. **ifstream name_file (“filename.dat”);**
   2. **iostream name_file (“filename.dat”);**
   3. **fstream name_file (“filename.dat”);**
3. При чтении данных из файла конец файла определяется функцией:
   1. **eol()** 2. **eof()** 3. **eok()**
4. Функция **eof()** возвращает значение 0, когда…

1. Конец файла еще не наступил;
2. Конец файла наступил;
3. Эта функция ничего не возвращает.

5. Для определения конца файла записан оператор **while (! name_file.eof()).** Цикл будет выполняться…
   1. Пока функция **eof()** возвращает ложь (0);
   2. Когда функция **eof()** возвращает истину (1);
   3. Такую конструкцию использовать нельзя.
6. При завершении работы с файлом его нужно закрыть функцией:
   1. **close();** 2. **close.file_name;** 3. **file_name.close().**
7. При чтении массивов или структур из файла используется функция:
   1. **read;** 2. **fread;** 3. **ifread.**
8. Из какого файла будет прочитана информация при выполнении оператора **ifstream koord ("koord1.dat");**?
   1. **koord**; 2. **koord1**; 3. Здесь имя файла не указано.
9. Функция **write** используется для…
   1. Ввода массива (структуры) в файл;
   2. Вывода массива (структуры) из файла;
   3. Вывода массива (структуры) на экран.