РАБОТА С ФАЙЛАМИ В С++

Библиотека `fstream` предоставляет функционал для считывания данных из файла и для записи в файл. В целом она очень похожа на `iostream`, которая работает с консолью, поскольку консоль это тоже файл.

`<fstream>`:

`<ifstream>` — файловый ввод ;

`<ofstream>` — файловый вывод

Наиболее частые операции:

Операторы перенаправления ввода\вывода –	`<<` и `>>`
Методы записи и чтения строк	`getline()` и `get()` с `put()`
Потоковая запись и чтение методами	`write()` и `read()`
Методы открытия\создания и закрытия файлов	`open()` и `close()`
Методы проверки открыт ли файл	`is_open()`
и достигнут ли конец файла	`eof()`
Настройка форматированного вывода для >> с помощью	`width()` и `precision()`
Операции позиционирования:	`tellg()`, `tellp()` и `seekg()`, `seekp()`

```
ofstream fout("file.txt", ios::app);
fout.open("file.txt", ios::app);
ios::out | ios::in - открытие файла для записи и чтения.
```

```
// текстовый файл
#include <fstream>
using namespace std;
int main()
{
  ofstream fout;
  fout.open("file.txt");
  fout << "Work with files in C++";
  fout.close();
  return 0;
}
```

Константа	Описание
`ios::in`	открыть файл для чтения
`ios::out`	открыть файл для записи
`ios::ate`	при открытии переместить указатель в конец файла
`ios::app`	открыть файл для записи в конец файла
`ios::trunc`	удалить содержимое файла, если он существует
`ios::binary`	открытие файла в двоичном режиме

Проверить, открылся ли файл можно двумя способами:

1. проверить переменную файла в логическом выражении
2. использовать метод `is_open()` :

```
#include <iostream>
#include <fstream>
using namespace std;
 int main()
{
setlocale(LC_ALL, "rus");
ifstream file (“Text.txt");
 if (!file)
{cout << "Файл не открыт\n\n";
return -1; }
else {
cout << "Все ОК! Файл открыт!\n\n";
return 1; }
}
```

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

int main()
{
setlocale(LC_ALL, "rus");
ifstream file (“file.txt");
if (file.is_open()) // вызов метода
cout << "Все ОК! Файл открыт!\n\n" << endl;
else
{ cout << "Файл не открыт!\n\n" << endl;
return -1; }  }
```

Считывание данных из файла

Для текстовых файлов оператор >> считает вещественное, целое и строку. Считывание строки закончится, если появится пробел или конец строки.

```
// чтение слов из файла
for(file>>s; !file.eof(); file >> s)
  cout << s << endl;
```

Считывание целой строки до перевода каретки производится так же как и в `iostream` методом `getline()` в виде функции, если считывается строка типа `string`.

```
//1 чтение строки из текста
string s;
getline(file,s);
cout << s << endl;
```

Если же читать нужно в массив символов char[], то либо get() либо getline() именно как методы:

```
// 2 чтение массива символов
int n = 10;
//Создаем буфер для чтения
char* buffer = new char[n+1];
        buffer[n]=0;
//Читаем n символов
file.get(buffer,n);
//Или так, но до первого пробела
file.getline(buffer,n,' ');
//выводим считанное
cout << buffer;
//Освобождаем буфер
delete [] buffer;
```

Создание и чтение текстового файла

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
 int main()
{ int i; string s;
ofstream file1 ("file.txt");
for (i=0; i<=5; i++)
{cin >> s;
file1 << s<<endl;  }
file1.close();
ifstream file2 ("file.txt");
for (; !file2.eof(); ) {
getline(file2, s);
cout << s << endl;  }
return 0; }
```

Текстовый файл содержит несколько строк, в каждой из которых записано единственное выражение вида:

a#b (без ошибок!), где

a, b - целочисленные величины,

- операция +, -, /, *.

```
// Вывести каждое из выражений и их значения.
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;
int main()  {
  long a, b;  char s[256], c;  int i;
  cout << "File name? "; cin >> s;
  ifstream f;  f.open(s);
  while (!f.eof())
   {    f.getline(s, 256);
    i=0; a=0;
    while (s[i]>='0'&&s[i]<='9')
     { a=a*10+s[i]-'0'; i++; }
    c=s[i++];    b=0;
    while (s[i]>='0' && s[i]<='9')
     {
         b=b*10+s[i]-'0';
         i++;
     }
    switch (c){
      case '+': a+=b; break;
      case '-': a-=b; break;
      case '/': a/=b; break;
      case '*': a*=b; break;}
      cout << s << " = " << a << endl;
    }
    f.close();
    cin.get();
    return 0;
}
/* В заданном файле целых чисел посчитать количество компонент,
кратных 3. */
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;
int main(){
  int r,ch;
  ifstream f;
  f.open("MyFile.txt");
  ch=0;
  for (; !f.eof();)
  {
     f>>r;
     cout << r << " ";
    if (r%3==0) ch++ ;
  }
   f.close();
  cout << endl << "Answer: " << ch;
  cin.get();
  return 0;
}
```

Табличное задание функции в файле
(использование форматирования)

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <cmath>
#include <string.h>
using namespace std;

double fun(double x);
int main()
{
  double a, b, h, x; char s1[]="C:/", s[20];
  cout << "Enter the beginning and end of the segment, step-tab: ";
  cin >> a >> b >> h;
  cout << "File name? "; cin >> s;
  strcat(s1, s);
  ofstream f;
  f.open(s1);
  for (x=a; x<=b; x+=h)
    {
      f.width(10); f << x;
      f.width(15); f << fun(x) << endl;
    }
  f.close();
  cin.get();
  return 0;
}

double fun(double x) { return x*x +5*x; }
```

БИНАРНЫЕ ФАЙЛЫ

Считывание из бинарного файла производить лучше всего с помощью метода read().

```
int n=10; //Количество байт для чтения из файла

//Создаем буфер
char* buffer=new char[n+1]; buffer[n]=0;
//Читаем в него байты
file.read(buffer,n);
//выводим их на экран
cout<<buffer;
delete [] buffer;
```

ЗАПИСЬ В ФАЙЛ ДАННЫХ ПРОИЗВОЛЬНОГО ТИПА

```
тип x;
// x переводим в строку байтов
(char*)&x  // указатель на начало строки
sizeof(x)  // размер в байтах

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

int main() {
  const char* FName = "C:/f1.txt";
  int x = 100;
  double y = 5.988;
  /*Запись в бинарный файл*/
  ofstream out(FName, ios::binary);
  out.write((char*)&x, sizeof(x));
  out.write((char*)&y, sizeof(y));
  out.close();

  /*Чтение из бинарного файла*/
  int x1 = 0;
  double y1 = 0;
  fstream in(FName,ios::binary);
  in.read((char*)&x1, sizeof(x));
  in.read((char*)&y1, sizeof(y));
  in.close();
  cout << x1 << '\n' << y1 << '\n';
  cin.get();
}

//  Чтение массива из бинарного файла
#include <fstream>
#include <iostream>
using namespace std;

int main() {
  const char* FName = "C:\\MyFiles\\text.txt";
  char S[255] = {};

  ifstream in(FName,ios::binary);
  in.read((char*)&S, sizeof(S));  // в случае с массивом можно так
                                  // in.read((char*)S, sizeof(S));

  in.close();
  cout << S << '\n';
  cin.get();
}
```

Запись и чтение структуры в бинарный файл

```
#include <string.h>
#include <iostream>
#include <fstream>
using namespace std;

struct Worker {
 char Name[25]; //Фамилия
 float salary;  //Зар. плата
 int age; }; //Возраст

int main()  {
  const char *FName="C:/Worker.bin"; //Путь к файлу
  Worker teacher; //объект teacher
  Worker w1;      //объект w1
  /*ЗАПОЛНЯЕМ СТРУКТУРУ*/
  strncpy(teacher.Name, "Pupkin", 25);
  teacher.age = 30;
  teacher.salary = 1523.99;

  /*ЗАПИСЫВАЕМ СТРУКТУРУ В ФАЙЛ*/
  ofstream f1(FName, ios::binary | ios::out);
  f1.write((char*)&teacher, sizeof(teacher));
  f1.close();
  /*ЧИТАЕМ СТРУКТУРУ ИЗ ФАЙЛА */
  ifstream f2(FName, ios::binary | ios::in);
  f2.read((char*)&w1, sizeof(teacher));
  f2.close();
  /*ВЫВОД ДАННЫХ НА ЭКРАН*/
  cout << w1.Name << '\t' << w1.age << '\t' << w1.salary << '\n';
}
```

ПРОИЗВОЛЬНЫЙ ДОСТУП К ФАЙЛУ

Система ввода-вывода C++ обрабатывает два указателя, ассоциированные с каждым файлом:

- `get pointer g` - определяет, где в файле будет производиться следующая ***операция ввода***;
- `put pointer p` - определяет, где именно в файле будет производиться следующая ***операция вывода.***

Операции ввода или вывода автоматически перемещают соответствующий файловый указатель.

С помощью методов `seekg()` и `seekp()` можно получить доступ к файлу в произвольном месте.

- `ifstream &seekg(смещение, позиция);`
- `ofstream &seekp(смещение, позиция);`

Смещение определяет область значений в пределах файла (long int).

Позиция смещения определяется как

Позиция	Значение
`ios::beg`	начало файла
`ios::cur`	текущее положение(по умолчанию)
`ios::end`	конец файла

```
file.seekg(0,ios::end);  //Стать в конец файла
file.seekg(10,ios::end); //Стать на 10 байтов с конца
file.seekg(30,ios::beg); //Стать на 31-й байт
file.seekg(3,ios::cur);  //перепрыгнуть через 3 байта
file.seekg(3); //перепрыгнуть через 3 байта - аналогично
```

Определение текущей позиции файлового указателя

`streampos tellg()` - позиция для ввода

`streampos tellp()` - позиция для вывода

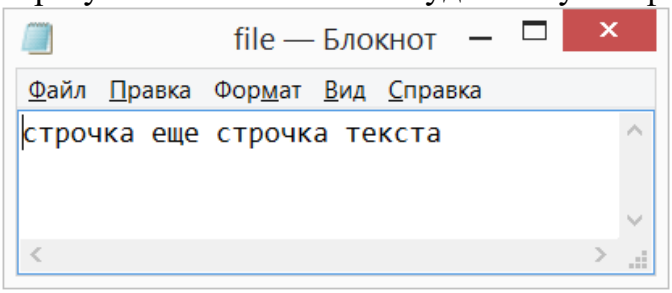
```
//становимся в конец файла
file.seekg(0, ios::end);
//Получаем текущую позицию
cout << "Размер файла (в байтах): " << file.tellg();

// чтение массива символов-байт с указанной позиции
// возвращает этот массив
char *myread (int position, int count){
  if(!f.is_open()) return 0;
  f.seekg(position);
  if(f.eof()) return 0;
  char *buffer=new char[count];
  f.read( buffer, count);
  return buffer;
}

#include <iostream>
#include <fstream>
```

```
using namespace std;
int main() {
  char s[80];
  fstream inout;
  inout.open("file.txt",ios::out);
  inout <<"строчка текста" <<endl;
  inout.seekp(8, ios::beg);
  inout << "еще строчка текста";
  inout.close();
  inout.open("file.txt", ios::in);
  inout.seekg(-6, ios::end);
  inout >> s;
  inout.close();
  cout << s;    cin.get();
  return 0;
}
```

В результате выполнения будет получен файл:



а на консоли получим