Организация ввода-вывода в C++

* Все средства ввода-вывода в языке C++ реализованы в виде библиотечных перегруженных операций << и >> (извлечения и вставки), манипуляторов, классов потоков, констант, глобальных переменных, функций и типов данных, находящихся в составе библиотеки управления потоками.
* Stream-библиотека выполнена в виде иерархии классов, которые описаны в нескольких заголовочных файлах.
* Подобно тому, как в файле iostream.h объявлены базовые средства ввода-вывода, обеспечивающие обмен данными между программой и стандартными устройствами ввода-вывода через потоки(streams), в файле fstream.h становятся доступны средства обмена данными между программой и файлами через потоки.

К основным вариантам организации ввода-вывода относятся:

* консольный форматированный ввод-вывод базовых типов;
* файловый форматированный ввод-вывод базовых типов;
* файловый символьный ввод-вывод;
* файловый строко-ориентированный ввод-вывод;
* форматированный обмен данными базовых типов со строкой в памяти.

Консольный форматированный ввод-вывод базовых типов осуществляется с помощью стандартных потоков cin и cout.

* Основные варианты организации ввода-вывода

Для обмена данными с файлами, размещенными на дисках, используются три стандартных класса файловых потоков: ***ifstream***, ***ofstream*** и ***iofstream***.

Для того чтобы начать работу с файлами, расположенными на логических дисках, необходимо «создать» соответствующие потоки.

*Файловый ввод-вывод

* Процесс работы с файлом через потоки можно разбить на 4 этапа:

*Создание потока (объявление переменной).

*Связывание потока с файлом и открытие (open) файла для работы в определенном режиме.

*Обмен данными с файлом через поток: запись в поток; чтение из потока; управление состоянием потока.

*Разрыв связи потока с файлом: освобождение буфера («флэширование» —flush),закрытие (close) файла и разрыв его связи с потоком.

При выполнении пунктов 2, 3, 4 следует контролировать наличие ошибок ввода-вывода средствами языка C++.

* При работе с потоками и файлами различают буферизированный (с использование буфера) и небуферизированный (без использования буферов) ввод-вывод.
* Буфер (buffer) представляет собой область оперативной памяти для промежуточного хранения данных, передаваемых между программой и внешним устройством.
* Вывод данных в поток с буфером приводит к выводу этих данных в соответствующий файл только после заполнения буфера.
* Вывод данных в небуферизованный поток приводит к немедленному выводу в файл.
* Символы извлекаются из потока методом getline() и помещаются в буфер. Число символов, помещаемых в буфер, задается вторым параметром функции getline(). При этом считывание происходит не более чем до первого встреченного символа-ограничителя «конец файловой строки», который представляется двумя символами **'\r' и '\n'.**
* Метод getiine()извлекает символ-ограничитель из потока, но не помещает его в буфер. Поэтому после каждого вывода содержимого буфера в поток fout вставляется и символ ограничитель. При использовании потока cin символ-ограничитель появляется в потоке при нажатии клавиши <Enter>.
* **cerr** – не буферизированный поток для стандартного вывода сообщений об ошибках (по умолчанию – на монитор);

Копирование одного файла в другой

```
#include <fstream.h>
#include <stdlib.h> // exit() // вспомогательная функция
void error (const char * message)
{
cerr <<'\n' << message <<'\n' ;
exit(1);
}
int main ()
{
char buf[128]; // вспомогательный буфер
ifstream fin; // 1. Создание потоков двух классов
ofstream fout;
fin.open("C:\\in.dat"); // 2. Открытие файлов в соответствующих режимах и связывание их с потоками
if (!fin.good()) error("Не могу открыть файл для чтения.");
fout.open("C:\\out.dat");
if (fout.fail()) error("Не могу открыть файл для записи.");
char ch; // 3. Выполнение операций обмена
while (fin.getline(buf, sizeof(buf))) fout << buf <<endl;
if (!fin.eof() || !fout.good())
error("Ошибка ввода-вывода при копировании файлов");
fin.close(); // 4. Закрытие файлов, разрыв связей потоков с файлами
fout.close();
return 0;
}
```

Рассмотрим программу копирования файла in.dat в файл out.dat. Предполагается, что файлы находятся в корневом каталоге логического диска C. Файл in.dat содержит строки символов, Длина файловой строки не превосходит 128 символов.

Для данного примера причинами выдачи сообщений «Не могу открыть файл...» могут быть следующие:

- отсутствие файла in.dat в корневом каталоге логического диска A или его особые атрибуты;
- неготовность устройства (не вставлена дискета, не закрыт карман и др.);
- у существующего файла out.dat установлен атрибут «только для чтения» или другие особые атрибуты.

Использование числовых значений

В файле param.dat записаны значения двух параметров программы в следующем виде:
5.6 // r
4.7 // h
Программа записывает в файл result.dat объем цилиндра, вычисленный на основе значений из файла param.dat.

```
#include <fstream.h>
#include <math.h>  // M_PI
int main()
{
char rbuf[81];
double r,h;
ifstream fin ("PARAM.DAT");
if (!fin.good())
{cerr << "\nНе могу открыть файл параметров.";
return 1;}
fin >> r; fin.getline(rbuf,sizeof(rbuf));
fin >> h; fin.getline(rbuf,sizeof(rbuf));
ofstream fout("RESULT.DAT");
fout << M_pi*r*r*h << " // v " << endl;
return 0;
}
```

* Переназначение стандартных потоков.

Приведенная программа осуществляет посимвольное копирование содержимого, входного файла в выходной файл при помощи методов write() и read() неформатированного ввода-вывода.

```
#include <fstream.h>
# include <stdlib.h>
ifstream fin;
ofstream fout;
int main()
{
char buf[1];
fin.open("input.dat");
if(fin.good()) cin = fin; // переназначение cin
else {cerr <<"Не могу открыть файл 'input.dat' ";
exit(l);}
fout.open("output.dat");
if(fout.good()) cout = fout; // переназначение cout
while(!cin.eof())
cin.read(buf, sizeof(buf));
cout.write(buf, sizeof (buf));
return 0;
}
```