ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Учебная практика
Основы C++

День 3
Работа с файлами

Потоки

Поток – последовательность байтов (символов), не зависящая от конкретных устройств, с которыми ведется обмен данными.

Поток (stream) – общий логический интерфейс с различными устройствами компьютера (Г. Шилдт)

Поведение потоков одинаковое, поэтому к ним применимы одни и те же функции и операторы ввода/вывода. Например, одни и те же методы применяются для вывода на экран, на принтер, записи в дисковый файл

Поток – последовательный логический интерфейс, который связан с физическим файлом (Г. Шилдт).

Под файлом подразумевается дисковый файл, экран, клавиатура, порт и т.д.

Потоки

Поток – связывается с файлом при выполнении операции открытия файла

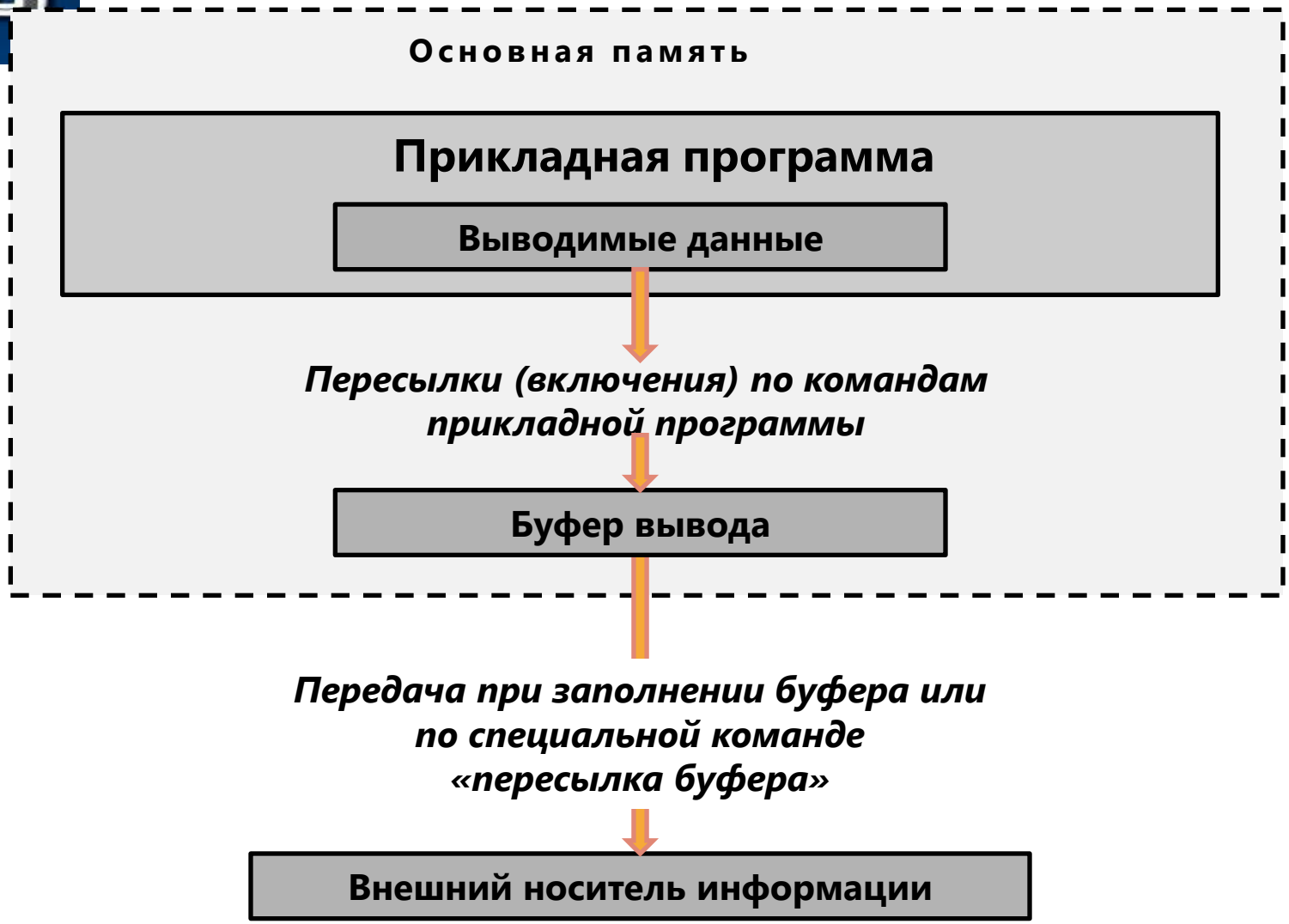
Поток отсоединяется от файла при выполнении операции закрытия файла

Существует два типа потоков

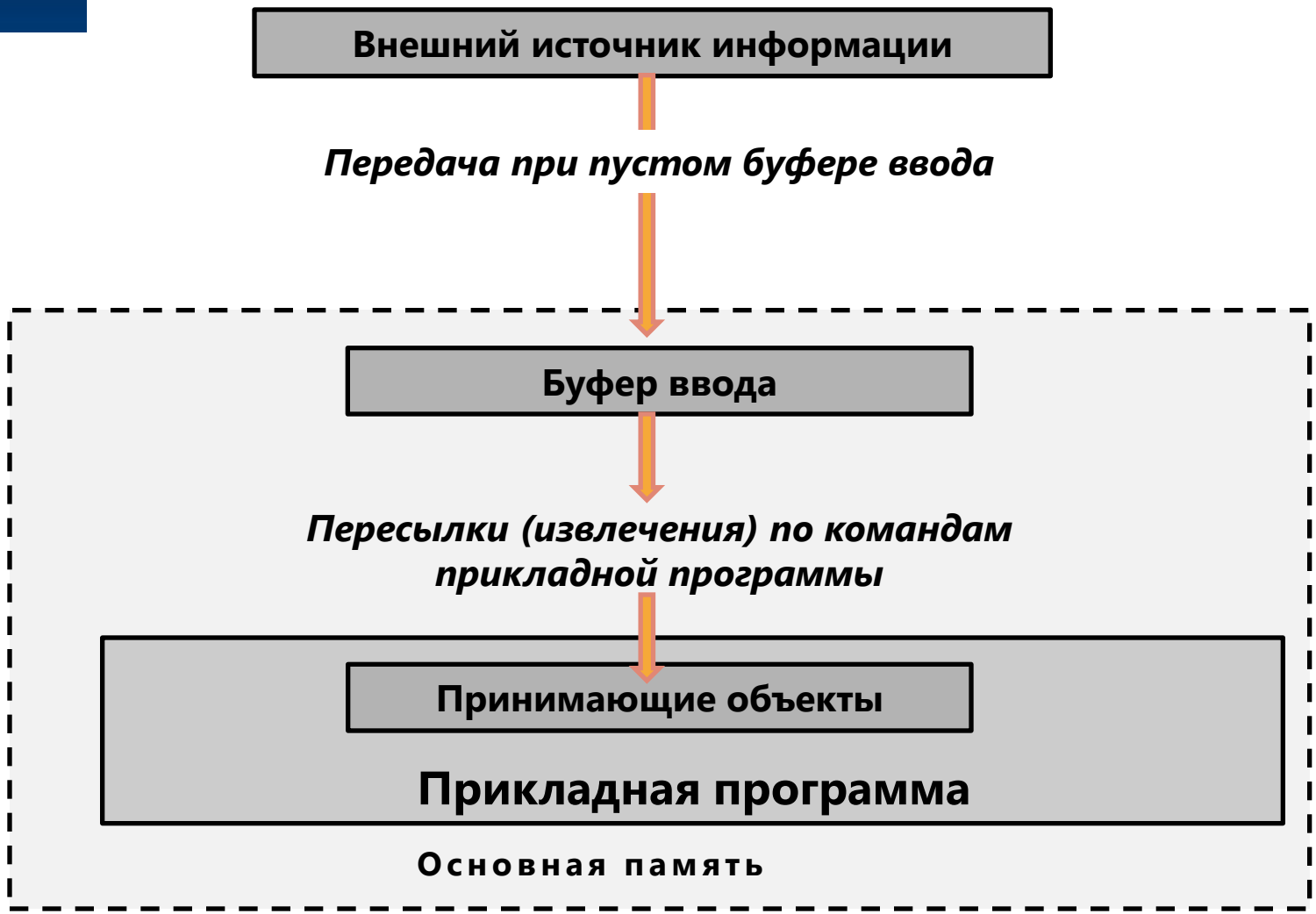
1. Текстовый (ввод/вывод символов), при этом могут происходить преобразования символов
2. Двоичный – преобразование не производится

Текущая позиция – та, с которой будет выполняться следующая операция доступа к файлу

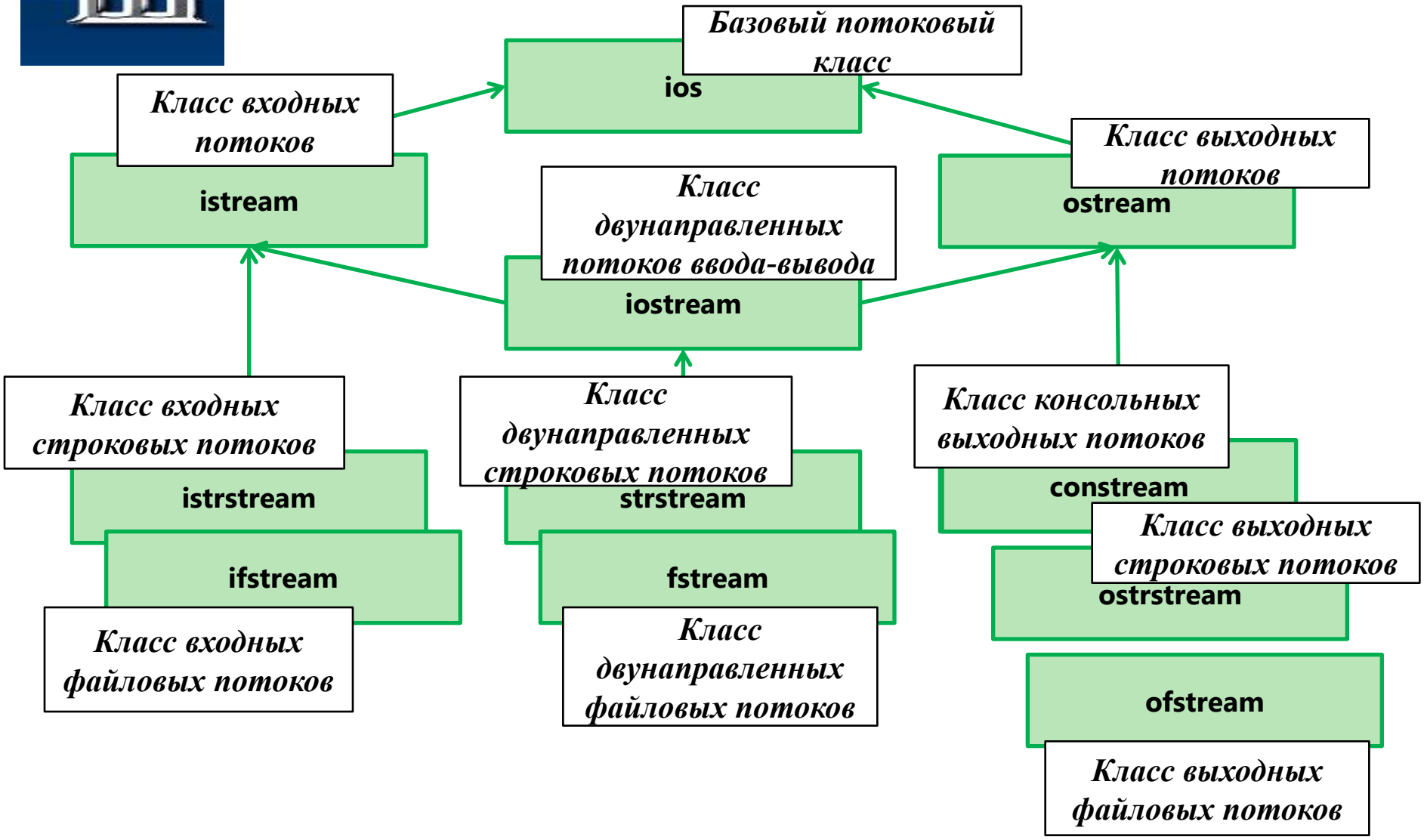
Буферизированный выходной поток



Буферизированный входной поток



Упрощенная схема иерархии классов библиотеки потокового ввода-вывода



Заголовочные файлы

Потоковые классы, их методы становятся доступны в программе, если в неё включены соответствующие заголовочные файлы.

```
#include <iostream>
```

Указанный в директиве заголовочный файл позволяет работать с библиотекой, содержащей средства для работы с потоками ввода-вывода (stream – поток, i – input, o – output). Здесь описаны классы ios, ostream, istream.

cin	объект класса **istream,** связанный со стандартным буферизированным входным потоком (как правило, клавиатура)
>>	операция класса **istream** «Извлечение данных из потока»
cout	объект класса **ostream**, связанный со стандартным буферезированным выходным потоком.
<<	операция класса **ostream** «Вставка данных в поток»

Заголовочные файлы

```
#include <fstream>
```

Указанный в директиве заголовочный файл позволяет работать с библиотекой, содержащей средства для работы с файловыми потоками. Позволяет работать с классами:

ofstream – для вывода данных в файл

ifstream – для ввода (чтения) данных из файла.

```
ofstream oFile; //Определили выходной файловый поток
ifstream inFile;//Определили входной файловый поток
```

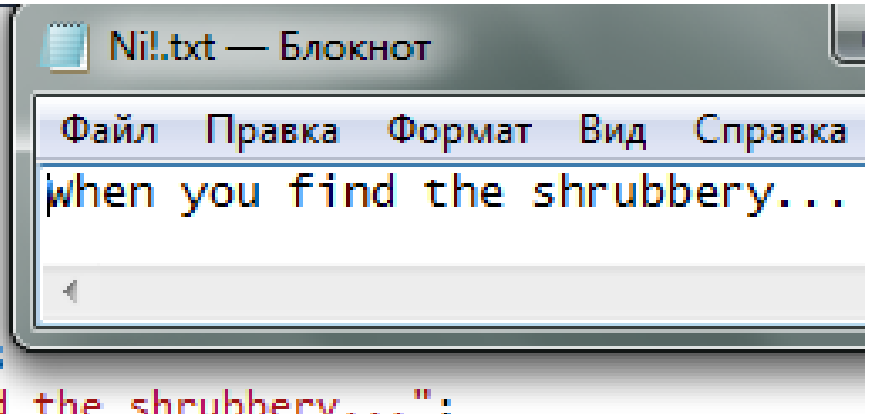
Файлы и потоки

```
#include <fstream>
```

- `ofstream`: Потоковый класс для записи в файлы
- `ifstream`: Потоковый класс для чтения из файлов
- `fstream`: Потоковый класс, с помощью которого можно осуществлять как запись в файлы, так и чтение из них

Пример программы для записи текста в файл

```
#include <iostream>
#include <fstream>
using namespace std;
void main()
{
  ofstream myfile;
  myfile.open ("Ni!.txt");
  myfile << "When you find the shrubbery...";
  myfile.close();
}
```



Соединение потока с конкретным файлом. Метод Open()

Для открытия файла необходимо использовать метод **open().**

- Первым параметром метода является имя файла,
- вторым параметром метода является режим открытия файла, или режим доступа

```
open(char *FileName, int <РежимДоступа>)
```

Соединяет файловый поток с конкретным файлом `FileName`.

Файл открывается в режиме доступа, переданном в параметре `<РежимДоступа>`.

Режимы открытия файла

Значение второго параметра метода open()	Результат
`ios::app`	При записи данные добавляются в конец файла, даже если текущая позиция была перед этим изменена
`ios::ate`	Указатель перемещается в конец файла. Данные записываются в текущую позицию (произвольное место) файла
`ios::in`	Поток создается для ввода данных в программу. Используемый для ввода файл сохраняется
`ios::out`	Поток создается для вывода данных из программы. Если файл уже существует, то он перезаписывается
`ios::nocreate`	Если файл не существует, возникают функциональные сбои;
`ios::noreplace`	Если файл уже существует, возникают функциональные сбои
`ios::binary`	Ввод-вывод данных будет выполняться в двоичном виде
`ios::out | ios::app`	Открывается существующий файл для добавления данных в конец файла
`ios::in | ios::out`	Открывается существующий файл для чтения и записи данных
`ios::in | ios::out | ios::app`	Открывается существующий файл для чтения и добавления данных в конец файла.

Метод eof()

Метод **eof()** возвращает 1, если указатель дошёл до конца файла.

Таким образом, считывание из файла можно проводить в цикле:

```
while (!file.eof()) {
      <тело_цикла>
}
```

Управление указателем чтения.

`seekg(offset, seekdir)`

Устанавливает смещение указателя чтения относительно параметра `seekdir`.

`seekdir` может принимать следующие значения:

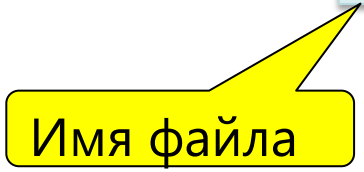
- `end` – конец файла
- `cur` – текущее положение
- `beg` – начало файла.

```
…
ifstream inFile;
…
inFile.seekg(0, ios::beg ); //переходим в начало файла
inFile.seekg(0, ios::end ); //переходим в конец файла
inFile.seekg(1, ios::cur ); //сдвигаемся на одну позицию вперед
```

Метод close()

После завершения работы с потоком не следует забывать его закрывать

```
output.close();
```



Вывод в файл – буферизирован. Сначала заполняется буфер, затем его содержимое «сбрасывается» в файл.

Метод `close()` сбрасывает содержимое (незаполненного) буфера в файл

Метод getline()

Метод **getline()** позволяет считать массив символов необходимого размера в массив **char** или в строку **string**.
Последним символом в массиве будет являться **‘\0’.**

Пример:

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

using namespace std;

void main() {
  fstream file;
  file.open("file.txt", ios::in);
  char str[5];
  file.getline(str, 5);
  cout<<str<<endl;
  file.close();
  system("pause");
}
```

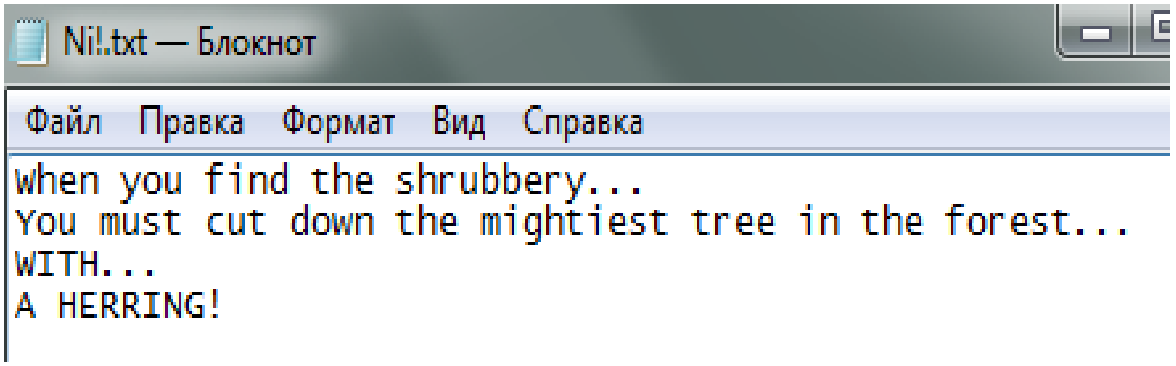
Результат исполнения программы при тексте в файле «Hello, World»

```
Hell
Для продолжения нажмите любую клавишу
```

Обратите внимание, что метод getline() с параметром 5 записывает в массив 5 символов: 4 символа из файла и символ ‘\0’

Построчное чтение текста из файла (ifstream)

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
void main()
{
  string line;
  ifstream myfile ("Ni!.txt");
  if (myfile.is_open())
  {
      while (!myfile.eof())
        {
          getline(myfile,line);
          cout << line << endl;
        }
    myfile.close();
  }
  else cout << "Unable to open file";
  system("pause");
}
```




```
When you find the shrubbery...
You must cut down the mightiest tree in the forest...
WITH...
A HERRING!
Для продолжения нажмите любую клавишу . . .
```

Чтение массива из файла с помощью fstream

```
1  #include<iostream>
2  #include<fstream>
3
4  using namespace std;
5
6  void main(){
7      int n;
8      int* arr;
9      fstream file;
10     file.open("file.txt", ios::in);
11     file>>n;
12     cout<<"Length: "<<n<<endl;
13     arr = new int[n];
14     for (int i = 0; i < n; i++) {
15         file>>arr[i];
16     }
17     cout<<"Array: ";
18     for (int i = 0; i < n; i++) {
19         cout<<arr[i]<<" ";
20     }
21     cout<<endl;
22     file.close();
23     delete [] arr;
24     system("pause");
25 }
```

В примере файл входных данных имеет следующий формат:

- первое число в файле – количество элементов массива;
- остальные числа – элементы массива.

Приведённая программа позволяет считывать из файла массив с числами и выводить его на экран.

Программа считывает массив вне зависимости от того, как числа разбиты на строки во входном файле.

```
3 -1 4 86
```

```
3
-1 4 86
```

```
3
-1
4
86
```

```
Length: 3
Array: -1 4 86
Для продолжения нажмите любую клавишу . . .
```

Пример 2. Чтение массива из файла.

```
int main()
{
	fstream fin;
	fin.open("text_matr.txt", ios::in);
	int n;
	int a[10][10];
	fin >>n;
	  for (int i=0;i<n;i++)  {
		for (int j=0;j<n;j++) {
			fin >>a[i][j];
cout<<a[i][j]<<" ";
			}
			cout<<endl;
	  }
	fin.close();
	system("PAUSE");
}
```

text_matr.txt

```
4
1 2 4 3
2 3 5 6
2 3 4 5
1 2 3 4
```

Работа с файлами

1) Подключить `#include <fstream>`

2) Открыть поток на вывод `ofstream fi;`

3) Связать с файлом на диске

```
fi.open("D:\\PI\\String.txt", ios::ate);
```

4) Пример вывода числа в файл с заданным количеством знаков после .

```
fi << fixed << setprecision(2) << vmin[i] << "  ";
```

5) Закрыть файл (поток) `fi.close();`

Пример. Запись в файл.

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

int main()
{
	ofstream fOut;
	fOut.open("text.txt", ios::app);
	fOut << “Hello!\n";
	fOut.close();
	system(“PAUSE”);
}
```

Пример записи в файл

```
ofstream fi;
  fi.open("D:\\BPI153\\String.txt",ios::ate);
  double dlin=0;
  cout << "dlin = " << dlin << '\n';
  fi << "Длина = " << dlin << '\n';
  for (int i = 0; i < 7; ++i)
       fi << (i * 5) << "  ";
  fi << '\n';
fi.close();
```