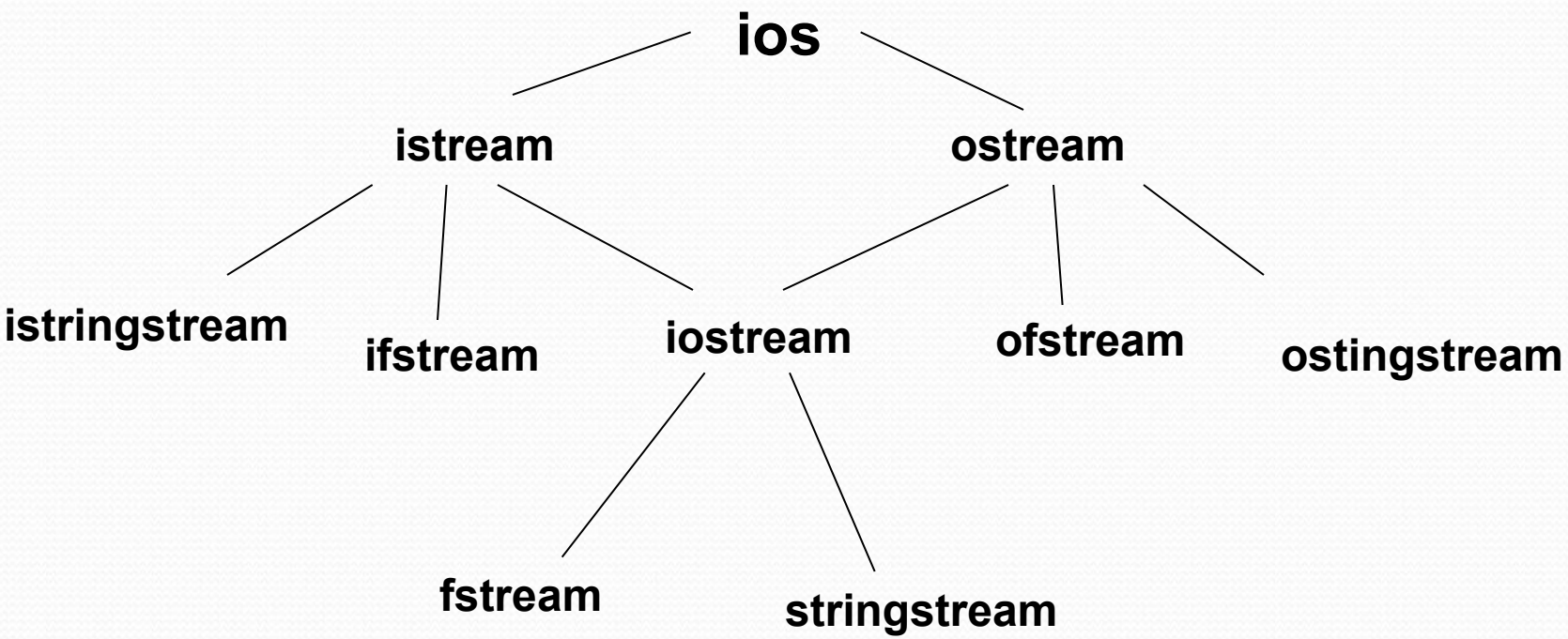
Программирование
Тема 6.1 Работа с файлами в C++

Классы потоков



- Базовые свойства всех потоков содержит класс ios, объявленный в файле iostream.h На слайде приведен фрагмент иерархии классов потоков .
- Если в программе требуется работать с файлами через потоки, то необходимо подключить файл fstream.h. Тогда можно создавать потоковые объекты трех классов: ifstream – для ввода из файлов; ofstream – для вывода в файл; fstream – для обмена с файлом в двух направлениях.
- Процесс работы с файлом через потоки включает 4 этапа:
  1. Создание потока;
  2. Связывание потока с файлом и открытие файла в определенном режиме;
  3. Обмен данными с файлом через поток;
  4. Разрыв связи потока с файлом.
- При работе со стандартными потоками действия этапов 1, 2 и 4 выполняются автоматически. По умолчанию при корректном завершении программы или при выходе из области видимости потока освобождение буфера и закрытие файла осуществляется автоматически. Тем не менее, этап 4 повышает надежность программ при работе с файлами.
- При выполнении этапов 2, 3, 4 следует контролировать наличие ошибок ввода-вывода.

Конструкторы классов файловых потоков:

```
ifstream( ); ofstream( ); fstream( );
ifstream(const char *name, int mode = ios::in);
ofstream(const char *name, int mode = ios::out| ios::trunc);
fstream(const char *name, int mode = ios::in | ios::out);
enum open_mode
{   in = 0x01, // Открыть для чтения
  out = 0x02, // Открыть для записи
  ate = 0x04, // Установить указатель на конец файла
  app = 0x08, // Открыть для добавления в конец
  trunc= 0x10, // Если файл существует, удалить
  nocreate= 0x20, // Если файл не существует, выдать ошибку
  noreplace= 0x40, // Если файл существует, выдать ошибку
  binary = 0x80,  // Открыть в двоичном режиме
}
```

Методы открытия и закрытия файловых потоков:

```
bool open(const char *name, int mode);
void close(void);
```

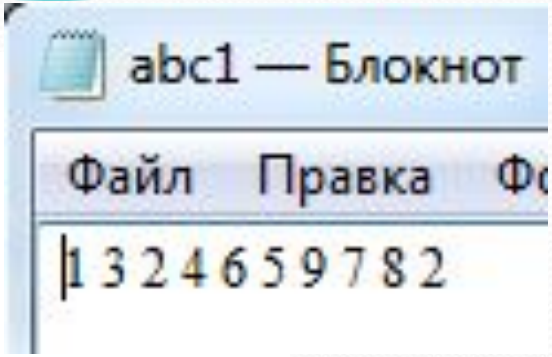
- Поток — это общий последовательный логический интерфейс с различными устройствами, входящими в состав компьютера (дисплей, клавиатура, принтер, жесткий диск и др.).
- Потоки, работающие с дисплеем и клавиатурой называют стандартными потоками, а потоки, работающие с физическим файлом, размещаемым или размещенным в дисковом пространстве компьютера (жестком диске) называют файловыми.
- Существуют два вида потоков (файлов): текстовый и двоичный (бинарный). Текстовый поток предназначен для ввода/вывода символьных данных, при этом могут происходить некоторые преобразования символов. Например, символ новой строки может быть преобразован при выводе в последовательность из двух символов: возврат каретки и переход на новую строку. В свою очередь при работе с бинарным потоком никакого преобразования символов не происходит. Текущая позиции или указатель потока (файла) – это место в потоке (файле), с которого будут выполняться операции доступа к компонентам потока.
- В C++ все операции, связанные с файловыми потоками, определены в fstream.h. Поэтому в разделе директив необходимо указать #include <fstream>.
- Каждый класс файловых потоков (ifstream, ofstream и fstream) содержит конструкторы, с помощью которых можно инициализировать объекты этих классов различными способами.
- Если потоковый объект создается с использованием конструктора с параметрами, то поток будет связан с файлом (файл будет открыт). Вторым параметром конструктора является режим открытия файла. Если установленное по умолчанию значение не устраивает программиста, можно указать другое, составив его из битовых масок, определенных в классе ios.
- Если потоковый объект создается с использованием конструктора без параметров, то поток не будет связан с файлом (файл не будет открыт). Открыть файл в этом случае можно с помощью метода open( ), имеющего такие же параметры, что и у конструктора с параметрами.
- Для закрытия потока определен метод close(), но поскольку он неявно выполняется деструктором, явный вызов необходим только тогда, когда требуется закрыть поток раньше конца его области видимости.

- **Чтение и запись данных текстовых файлов:**
  - операции `>>` и `<<` ;
  - метод `getline(char [] buf, int n);`
  - функция `getline(istream in, string str);`

- **Считывание и запись в файл блоков данных:**
  - `istream &read(char *buf, streamsize num);`
  - `ostream &write(const char *buf, streamsize num);`

- При чтении или записи данных в текстовый файл можно использовать операции << и >>. Например, для чтения целочисленной переменной x из файлового потока in необходимо использовать оператор: in>>x; а для записи в выходной файловый поток out значения цело численной переменной x: out<<x.
- Для чтения из файлового потока строк можно использовать операцию >>, но у нее есть недостаток – она вводит строку до первого пробела.
- Для чтения из файлового потока строк, содержащих пробелы, нужно использовать или метод getline() или специальную функцию getline().
- Метод getline() используется для ввода строк, которые являются массивами символов с завершающем символом ‘\0’. Метод getline() считывает из входного потока (n-1) символов или менее, если символ перевода строки встретился раньше, и записывает их в строковую переменную s. Символ перевода строки также считывается (удаляется) из входного потока. Вместо символа перевода строки в строковой переменной размещается завершающий 0.
- Функция getline() используется для ввода строк, которые являются объектами класса string.
- Функция read() считывает num байт данных из связанного с файлом потока в буфер buf. Функция write() записывает num байт данных в связанный с файлом поток из буфера buf. Тип streamsize является разновидностью целого типа. Он позволяет хранить самое большое количество байтов, которое может быть передано в процессе любой операции ввода/вывода.

Пример 1: Чтение элементов одномерного массива из текстового файла и запить в текстовый файл

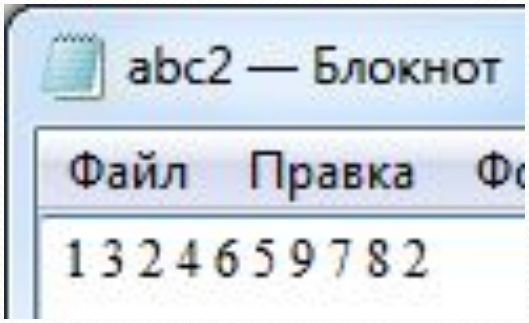



```
#include <iostream>
#include <fstream>
using namespace std;
int main(void)
{   int n = 0; int a[10];
    ifstream f; // объявление потока для чтения
    f.open("E:\\abc1.txt");  // режим ios::in
    if (f) {  cout << "Read file ! \n";
              while(!f.eof())
               { f >> a[n]; // чтение из файла
                 n++;
               } f.close();
               for(int i=0;i < n;i++)
               cout << a[i] << " ";
          }
    else cout << "File not found \n";
```

```
Read file !
1 3 2 4 6 5 9 7 8 2
File write!_
```

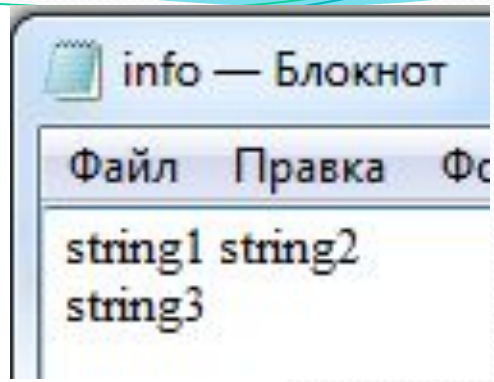
Пример 1: Чтение элементов одномерного массива из текстового файла и запить в текстовый файл

```
ofstream f2; // объявление потока для записи
    f2.open("E:\\abc2.txt"); // режим ios::out
     if (f2)
       { for(int i = 0;i < n;i++)
            f2 << a[i] << " "; // запись в файл
         cout << "\nFile write!";
          f2.close();
         }
     else cout << "File not found \n";
}
```



Пример 2: Вывод на экран содержимого текстового файла

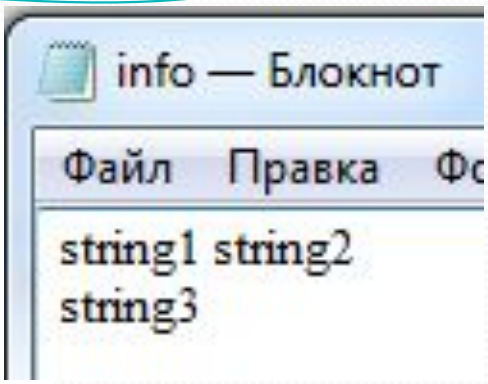
```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{ char buf[81];
  ifstream f("E:\\info.txt");
  if (!f)
       { cout << "Ошибка открытия файла";
         return 1;
       }
  while (!f.eof())
   { f.getline(buf, 81);
     cout << buf << endl;
   }
  return 0;
}
```




```
string1 string2
string3
```

Пример 3: Вывод на экран содержимого текстового файла

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

int main()
{ string buf;
  ifstream f("E:\\info.txt");
  if (!f)
       { cout << "Ошибка открытия файла";
         return 1;
       }
  while (!f.eof())
   { getline(f, buf);
     cout << buf << endl;
   }
  return 0;
}
```




```
string1 string2
string3
```

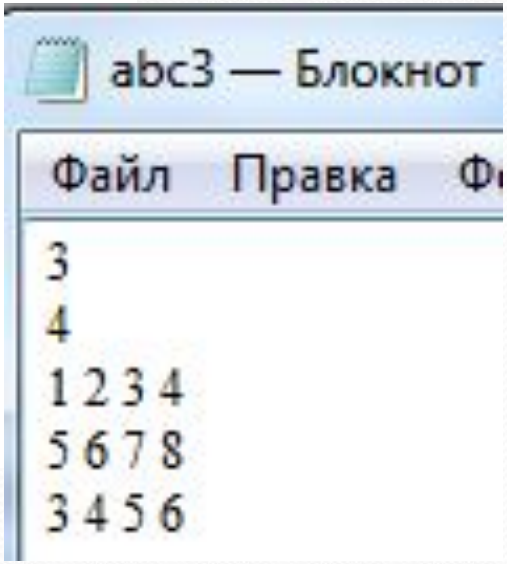
Пример 4: Запить в текстовый файл элементов двумерного массива

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

int main(void)
{ int nrow, ncol; int i,j;
  int **a;
  cout << " nrow = "; cin >> nrow;
  cout << " ncol = "; cin >> ncol;
  cout << "\n";
  // Выделение памяти под массив
  a = new int *[nrow];
  for(i = 0;i < nrow;i++)
       a[i]= new int [ncol];
 // Заполнение массива
  for(i = 0;i < nrow; i++)
   for(j = 0;j < ncol; j++)
        cin >> a[i][j];
```

```
 nrow = 3
 ncol = 4

1 2 3 4
5 6 7 8
3 4 5 6

File write!
```

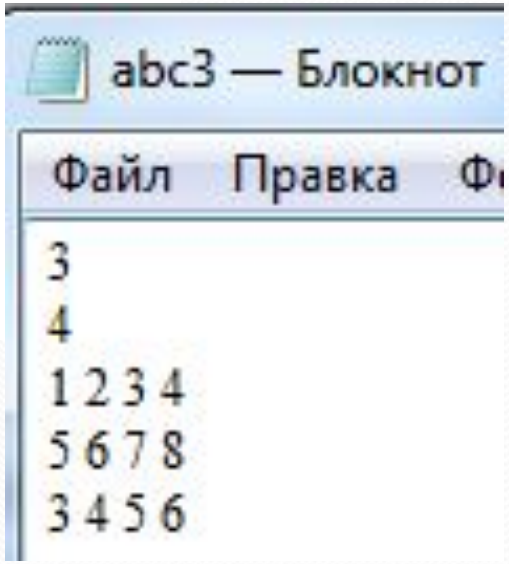
Пример 4: Запить в текстовый файл элементов двумерного массива

```
ofstream f2; // объявление потока для записи
    f2.open("E:\\abc3.txt"); // режим ios::out
      if (f2)
       {  f2 << nrow <<  "\n" << ncol <<  "\n";
          for(i = 0;i < nrow;i++)
            { for(j = 0;j < ncol;j++)
                 f2 << a[i][j]<< " ";
              f2 <<  "\n";
            }
        cout << "\nFile write!";
       f2.close();

       }
      else
       cout << "File not found \n";
 cout <<  "\n";
}
```



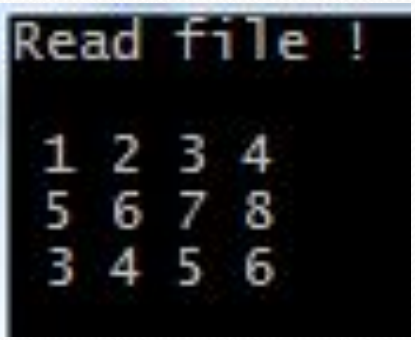
Пример 5: Чтение из текстового файла элементов двумерного массива

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

int main(void)
{ int nrow, ncol; int i,j;
  int **a;
  ifstream f; // объявление потока для чтения
  f.open("E:\\abc3.txt");  // режим ios::in
  if (f)
    { cout << "Read file ! \n";
        f >> nrow;   f >> ncol;
        // Выделение памяти под массив
        a = new int *[nrow];
        for(i = 0;i < nrow; i++)
           a[i]= new int [ncol];
```



Пример 5: Чтение из текстового файла элементов двумерного массива

```
        // Заполнение массива
            for(i = 0; i < nrow; i++)
              for(j = 0;j < ncol; j++)
                 f >> a[i][j];
              cout << "\n";
          f.close();

        // вывод  массива
            for (i = 0;i < nrow; i++)
              { for (j = 0;j < ncol; j++)
                    cout << " " << a[i][j];
                cout <<  "\n";
              }
      }
    else cout << "File not found \n";
}
```



Пример 6: Запись и чтение бинарного файла

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{    ofstream f("E:\\Af.bin",ios::binary);
     int a,c;
     cout << "Введите 10 чисел через пробел:\n";
     for(int i=0;i<10;i++)
     {  cin >> a;
        f.write((char *)&a, sizeof(int));
     }
     f.close();
     cout<<"Введите число: "; cin >> c;
     ifstream myin("E:\\Af.bin",ios::binary);
     if (myin)
      { bool found = false;
             myin.read((char *)&a, sizeof(int));
             while (!myin.eof() && !found)
             { if (a == c) found = true;
                else
                  myin.read((char *)&a, sizeof(int));
             }
       if (found) cout << "Yes "<< endl;
          else cout << "No" << endl;
        myin.close();
       } else cout << "File not found!";
   return 0;
}
```

```
Введите 10 чисел через пробел:
1 2 3 4 5 6 7 8 9 10
Введите число: 4
Yes
```

Пример 7: Запись объектов в бинарный файл и чтение объектов из бинарного файла

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

class person
{ private:
    char name[40];    // фио
    int age;          // возраст
  public:
    person(){}
    ~person(){}
   void getData(void);     // ввод данных
   void showData(void);    // вывод данных

   void diskOut(ofstream& fout); // запись в файл
   void diskIn(ifstream& fin);  // чтение из файла
 };
```

Пример 7: Запись объектов в бинарный файл и чтение объектов из бинарного файла

```
void person::getData(void)     // ввод данных
   { cout << "Введите ФИО: "; cin.getline(name,40);
     cout << "Введите возраст: "; cin >> age; cin.get();
   }

void person::showData(void)    // вывод данных
  { cout << "\n ФИО: "  << name;
    cout << "\n Возраст: "  << age;
    cout << endl;
  }

 void person::diskOut(ofstream& fout)
   { fout.write((char*)this,sizeof(*this)); }

 void person::diskIn(ifstream& fin)
   { fin.read((char*)this,sizeof(*this)); }
```

Пример 7: Запись объектов в бинарный файл и чтение объектов из бинарного файла

```
int main()
{  person *a = new person[2];
    for (int i=0;i<2;i++)
      a[i].getData();

    ofstream fout("E:\\Person.bin",ios::binary);
    for (int i = 0; i < 2; i++)
        a[i].diskOut(fout);
    fout.close();

  ifstream fin("E:\\Person.bin",ios::binary);
    if (!fin) cout << "Error!" << endl;
     else
     { person temp;
        for (int i = 0; i < 2; i++)
         {  temp.diskIn(fin);
             temp.showData();
          }
        fin.close();
    }
    return 0;
}
```

```
Введите ФИО: Иванов Иван Иванович
Введите возраст: 20
Введите ФИО: Петров Петр Петрович
Введите возраст: 25

 ФИО: Иванов Иван Иванович
 Возраст: 20

 ФИО: Петров Петр Петрович
 Возраст: 25
```

Пример 8: Запись объектов в текстовый файл и чтение объектов из текстового файла

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

class person
{ private:
     string name;    // фио
     int age;        // возраст
  public:
     person(){}
     ~person(){}
    void getData(void);     // ввод данных
    void showData(void);    // вывод данных

    void diskOut(ofstream& fout); // запись в файл
    void diskIn(ifstream& fin);  // чтение из файла
 };
```

Пример 8: Запись объектов в текстовый файл и чтение объектов из текстового файла

```
void person::getData(void)     // ввод данных
  { cout << "Введите ФИО: "; getline(cin, name);
    cout << "Введите возраст: "; cin >> age; cin.get();
  }

void person::showData(void)    // вывод данных
 { cout << "\n ФИО:"  << name;
   cout << "\n Возраст:"  << age;
   cout << endl;
 }

void person::diskOut(ofstream& fout)
  {    fout << name << endl;
       fout << age << endl;
  }

void person::diskIn(ifstream& fin)
  {    getline(fin, name);
       fin >> age; fin.get();
  }
```

Пример 8: Запись объектов в текстовый файл и чтение объектов из текстового файла

```
int main()
{  person *a = new person[2];
    for (int i=0;i<2;i++)
      a[i].getData();

    ofstream fout("E:\\Person.txt");
    for (int i = 0; i < 2; i++)
        a[i].diskOut(fout);
    fout.close();

  ifstream fin("E:\\Person.txt");
    if (!fin) cout << "Error!" << endl;
     else
     { person temp;
        for (int i = 0; i < 2; i++)
         {  temp.diskIn(fin);
             temp.showData();
          }
        fin.close();
    }
    return 0;
}
```

```
Введите ФИО: Иванов Иван Иванович
Введите возраст: 23
Введите ФИО: Сидоров Иван Петрович
Введите возраст: 18

ФИО:Иванов Иван Иванович
Возраст:23

ФИО:Сидоров Иван Петрович
Возраст:18
```

Контрольные вопросы

1. Понятие потока. Классы файловых потоков C++. Понятие текстового потока. Примеры чтения и записи данных текстовых файлов.
2. Понятие бинарного потока. Примеры чтения и записи данных бинарных файлов.