Лекция 5

Файлы

Потоки и файлы

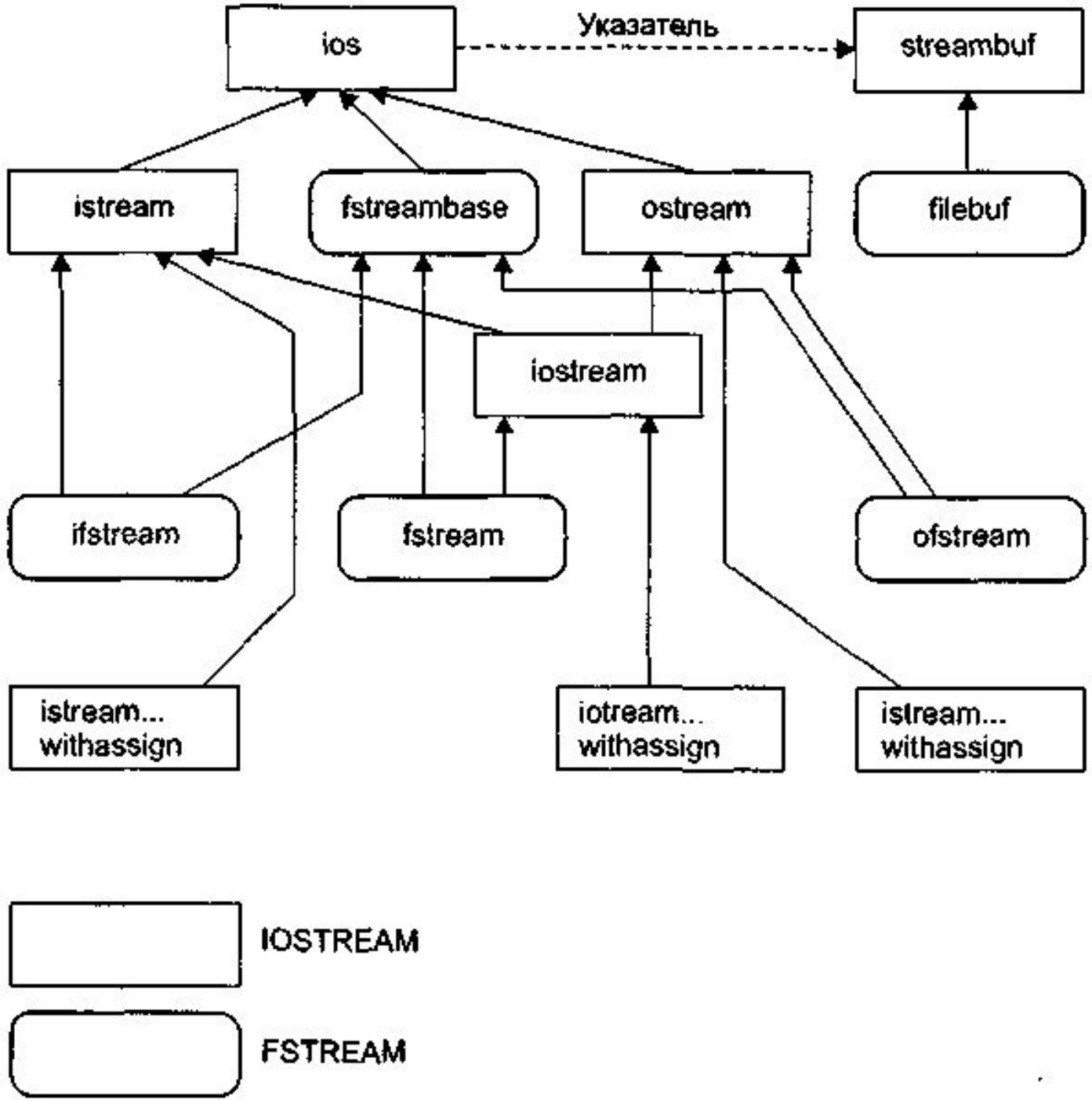
- Потоки
- Потоковый ввод/вывод дисковых файлов
- Указатели файлов
- Обработка ошибок файлового ввода/вывода

Потоки

Поток — это общее название потока данных.

cin
cout

для ввода/вывода вместо традиционных функций С для файлов —fscanf(),. fprintf()..


ios
Указатель
streambuf
istream
fstreambase
ostream
filebuf
iostream
ifstream
fstream
ofstream
istream...
withassign
iotream...
withassign
istream...
withassign
IOSTREAM
FSTREAM

ios

Таблица . Флаги форматирования класса ios

Флаг	Значение
skipws	Пропуск пробелов при вводе
left	Выравнивание по левому краю
right	Выравнивание по правому краю
internal	Заполнение между знаком или основанием числа и самим числом
dec	Перевод в десятичную форму
oct	Перевод в восьмеричную форму
hex	Перевод в шестнадцатеричную форму
boolalpha	Перевод логического 0 и 1 соответственно в «false» и «true»
showbase	Выводить индикатор основания системы счисления (0 для восьмеричной, для шестнадцатеричной)
showpoint	Показывать десятичную точку при выводе
uppercase	Переводить в верхний регистр буквы X, E и буквы шестнадцатеричной системы счисления (ABCDEF) (по умолчанию — в нижнем регистре)
showpos	Показывать «+» перед положительными целыми числами
scientific	Экспоненциальный вывод чисел с плавающей запятой
fixed	Фиксированный вывод чисел с плавающей запятой
unitbuf	Сброс потоков после вставки
stdio	сброс stdout, sterror после вставки

Все без исключения флаги могут быть выставлены с помощью методов setf() и unsetf(). Пример:

```
cout.setf ( ios::left ):   //выравнивание текста по левому краю
cout >> "Этот текст выровнен по левому краю"
cout.unsetf ( ios:: left ):  //вернуться к прежнему форматированию
```

Таблица 12.2. Манипуляторы **ios** без аргументов

Манипулято	Назначение
ws	Включает пропуск пробелов при вводе
dec	Перевод в десятичную форму
oct	Перевод в восьмеричную форму
hex	Перевод в шестнадцатеричную форму
endl	Вставка разделителя строк и очистка выходного
ends	Вставка символа отсутствия информации для
flush	Очистка выходного потока
lock	Закрыть дескриптор файла
unlock	Открыть дескриптор файла

Таблица Манипуляторы ios с аргументами

Манипулятор	**Аргумент**	Назначение
setw()	ширина поля (int)	Устанавливает ширину поля для вывода данных
setfill()	символ заполнения (int)	Устанавливает символ заполнения (по умолчанию, пробел)
setprecision()	точность (int)	Устанавливает точность(число выводимых знаков)
setiosflags()	Флаги форматирования (long)	Устанавливает указанные флаги форматирования
resetiosflagsf)	Флаги форматирования (long)	Сбрасывает указанные флаги форматирования

Таблица 4. Функции ios

Функция	Назначение
ch=fill();	Возвращает символ заполнения (символ, которым заполняется неиспользуемая часть текстового поля; по умолчанию — пробел)
fill(ch);	Устанавливает символ заполнения
p=precision();	Возвращает значение точности (число выводимых знаков для формата с плавающей запятой)
precision(p);	Устанавливает точность p
w=width();	Возвращает текущее значение ширины поля (в символах)
width(w);	Устанавливает ширину текущего поля
setf(flags);	Устанавливает флаг форматирования (например, ios::left)
unsetf(flags);	Сбрасывает указанный флаг форматирования
setf(flags, field);	Очищает поле, затем устанавливает флаги форматирования

Потоковый ввод/вывод дисковых файлов

Классы ifstream, ofstream и fstream объявлены в файле FSTREAM.

Старые функции языка С, такие, как fread() и fwrite() в С++ работают,

Запись данных

// Форматированный вывод в файл с использованием <<

```
#include <fstream>        // для файлового  ввода  вывода
#include <iostream>
#include <string>
using namespace std;
int main()   {
  char ch = 'x';
  int j = 77;
  double d = 6.02;
  string str1 = "Kafka";    // строки без пробелов
  string str2 = "Proust";
```

```
ofstream outfile("fdata.txt"); //создать объект ofstream
  outfile << ch           // вставить //(записать) данные
          << j
          << ' '          // пробелы между числами
          << d
          << str1
          << ' '          // пробелы между строками
          << str2;
  cout << "Файл записан\n";

  return 0;
  }
```

Чтение данных

```
// форматированное чтение из файла с помощью   >>
#include <fstream>            // для файлового ввода/вывода
#include <iostream>
#include <string>
using namespace std;
int main()   {
  char ch;   int j;    double d;
  string str1;   string str2;
  ifstream infile ( "fdata.txt“ ); // создать объект ifstream
  // извлечь (прочесть) из него данные
  infile >> ch >> j >> d >> str1 >> str2;
  cout << ch << endl            // вывести данные
       << j << endl
       << d << endl
       << str1 << endl
       << str2 << endl;
  return 0;
  }
```

Строки с пробелами

```
// файловый вывод строк с пробелами
#include <fstream>       // для операций
                         // файлового ввода/вывода
using namespace std;
int main()   {
   ofstream outfile("TEST.TXT"); // создать выходной файл
   // отправить текст в файл
   outfile << "Приходит март. Я сызнова служу.\n";
   outfile << "В несчастливом кружении событий \n";
   outfile << "изменчивую прелесть нахожу \n";
   outfile << "в смешеньи незначительных наитий.\n";
   return 0;
  }
```

getline()

```
// Файловый ввод (извлечение из файла) строк
#include <fstream>              // для файловых функций
#include <iostream>
using namespace std;

int main()   {
  const int MAX = 80;             // размер буфера
  char buffer [MAX];                // буфер символов
  ifstream infile ("TEST.TXT"); // создать входной файл

  while( !infile.eof() )              // цикл до EOF
    {
    infile.getline(buffer, MAX); // читает строку текста
    cout << buffer << endl;      // и выводит ее
    }
  return 0;
 }
```

Ввод/вывод символов

```
// Посимвольный файловый вывод

#include <fstream>                  // для файловых функций
#include <iostream>
#include <string>
using namespace std;

int main()   {
  string str = "Время – великий учитель, но, увы, "
               "оно убивает своих учеников.  Берлиоз";
  ofstream outfile("TEST.TXT");    // Создать выходной файл
  for(int j=0; j<str.size(); j++)          // каждый символ
    outfile.put( str[j] );                 // записывать в файл
  cout << "Файл записан\n";
  return 0;
  }
```

```
// Посимвольный файловый ввод

#include <fstream>              // для файловых функций
#include <iostream>
using namespace std;
int main()     {
  char ch;                     // символ для считывания
  ifstream infile("TEST.TXT"); // входной файл
  while( infile )              // читать до EOF или ошибки
    {
    infile.get(ch);            // считать символ
    cout << ch;                // и вывести его
    }
  cout << endl;
  return 0;
  }
```

Двоичный ввод/вывод

```
// Двоичный ввод/вывод целочисленных данных

#include <fstream>          // для файловых потоков
#include <iostream>
using namespace std;

const int MAX = 100;        // размер буфера
int buff[MAX];              // буфер для целых чисел
int main()   {
  for ( int j=0; j<MAX; j++ ) // заполнить буфер данными
    buff[j] = j;              // (0, 1, 2, ...)
  // создать выходной поток
  ofstream os("edata.dat", ios::binary);
  // записать в него
  os.write(reinterpret_cast<char*>(buff), MAX*sizeof(int) );
  os.close();               // должен закрыть его
```

```
for(j=0; j<MAX; j++)     // стереть буфер
      buff[j] = 0;

  // создать входной поток
  ifstream is("edata.dat", ios::binary);

   // читать из него
  is.read( reinterpret_cast<char*>(buff), MAX*sizeof(int) );

  for(j=0; j<MAX; j++)     // проверка данных
       if ( buff[j] != j )
           { cerr << "Некорректные данные!\n"; return 1; }
  cout << "Данные корректны\n";

return 0;
  }
```

При работе с бинарными данными в качестве второго параметра **write() и read()** следует использовать **ios::binary**

Оператор reinterpret_cast

Закрытие файлов

close()

Открытие файлов

Функция open()

Таблица 12.10. Биты режимов

Бит режима	Результат
in	Открытие для чтения (по умолчанию для ifstream)
out	Открытие для записи (по умолчанию для ofstream)
ate	Чтение, начиная с конца файла (AT End)
app	Запись, начиная с конца файла (AAPend)
trunc	Обрезать файл до нулевой длины, если он уже существует
nocreate	Не открывать несуществующий файл
noreplace	Не открывать для вывода существующий файл, если не установлены ate или app
binary	Открыть в бинарном (не текстовом) режиме

Указатели файлов

Функции **seekg()** и **tellg()** позволяют устанавливать и проверять текущий указатель чтения, а функции **seekp()** и **tellp()—** выполнять те же действия для указателя записи.

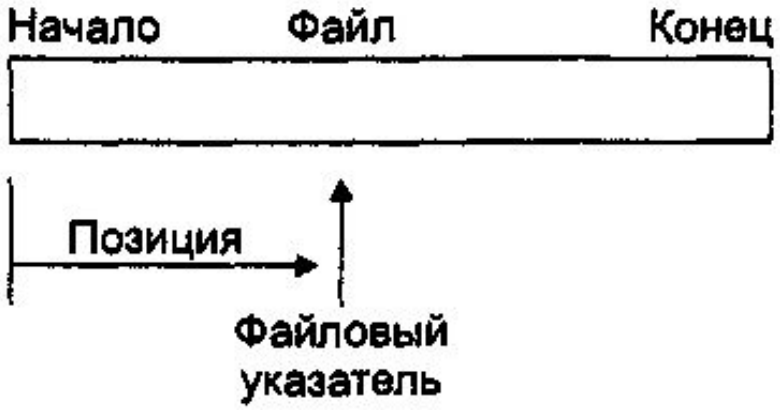


Рис. Функция seekg() с одним аргументом

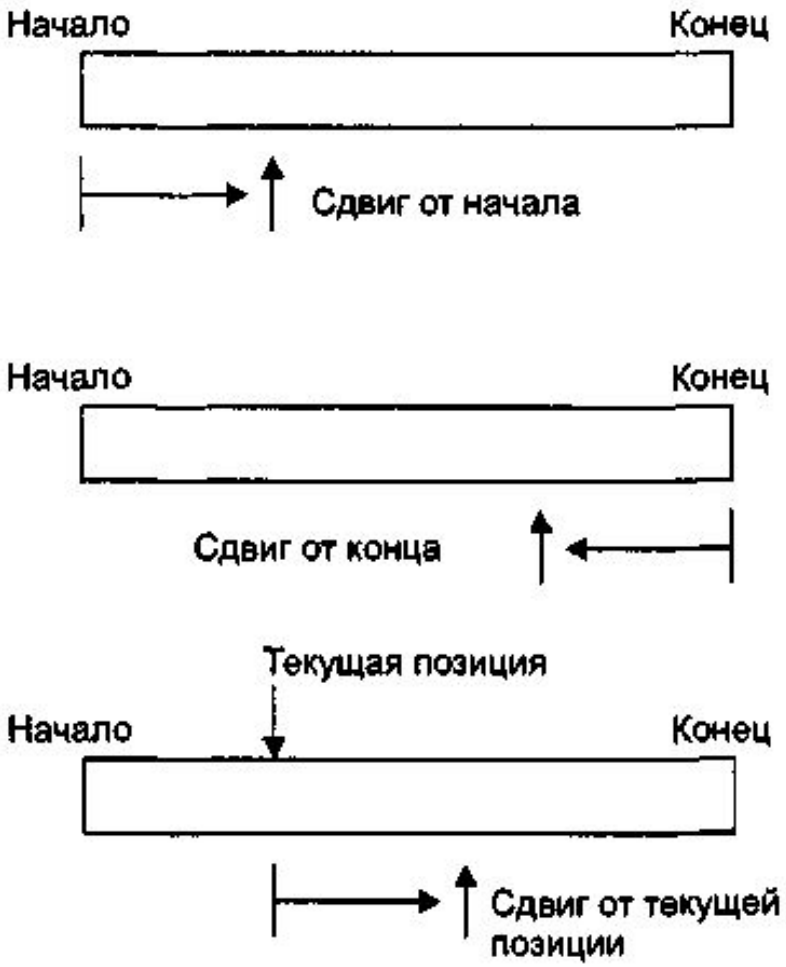


Рис. 12.5. Функция seekp() с двумя аргументами

```
// seekg.cpp  Поиск конкретного объекта в файле
#include <fstream>        // для файловых потоков
#include <iostream>
using namespace std;
person      {        // класс person
    protected:
    char name[80];        // имя человека
    int age;              // его возраст
   public:
    void getData()  {      // получить данные о человеке
          cout << "\n  Введите имя: "; cin >> name;
         cout << "  Введите возраст: "; cin >> age;
      }
    void showData(void)  { // вывод данных о человеке
         cout << "\n  Имя: " << name;
         cout << "\n  Возраст: " << age;
      }
 };
```

```
int main()   {
  person pers;                  // создать объект person
  ifstream infile;              // создать входной файл
  infile.open("GROUP.DAT",ios::in|ios::binary); // открыть файл

   infile.seekg(0, ios::end);    // установить указатель на 0
                                 // байт от конца файла
  int endposition = infile.tellg();      // найти позицию
  int n = endposition / sizeof(person);  // число человек
  cout << "\nВ файле " << n << " человек(а)";
  cout << "\nВведите номер персоны: ";
  cin >> n;
  int position = (n-1) * sizeof(person); // умножить размер
                       // данных под персону на число персон
  infile.seekg(position);               // число байт от начала
                       // прочитать одну персону
  infile.read( reinterpret_cast<char*>(&pers), sizeof(pers) );
  pers.showData();                      //вывести одну персону
  cout << endl;
  return 0;  }
```

```
// Обработка ошибок ввода/вывода
#include <fstream>            // для файловых потоков
#include <iostream>
using namespace std;
#include <process.h>          // для exit()
const int MAX = 1000;
int buff[MAX];

int main()   {
  for (int j=0; j<MAX; j++)    // заполнить буфер данными
        buff[j] = j;
  ofstream os;                 // создать выходной поток
    // открыть его
  os.open (“c:\\temp\\edata.dat", ios::trunc | ios::binary);
  if(!os)
        { cerr << "Невозможно открыть выходной файл\n"; exit(1); }
  cout << "Идет запись...\n"; // записать в него содержимое
                               // буфера
```

```
os.write( reinterpret_cast<char*>(buff), MAX*sizeof(int) );
  if(!os)       { cerr << "Запись в файл невозможна\n"; exit(1); }
  os.close();                  // надо закрыть поток
  for(j=0; j<MAX; j++)         // очистить буфер
    buff[j] = 0;
  ifstream is;                 // создать входной поток
  is.open(" c:\\temp\\ edata.dat", ios::binary);
  if(!is)
    { cerr << "Невозможно открыть входной файл\n";exit(1); }
  cout << "Идет чтение...\n"; // чтение файла

is.read( reinterpret_cast<char*>(buff), MAX*sizeof(int) );
  if(!is)
    { cerr << "Невозможно чтение файла\n"; exit(1); }
  for(j=0; j<MAX; j++)         // проверять //данные
    if( buff[j] != j )
      { cerr << "\nДанные некорректны\n"; exit(1); }
  cout << "Данные в порядке\n";
  return 0;    }
```