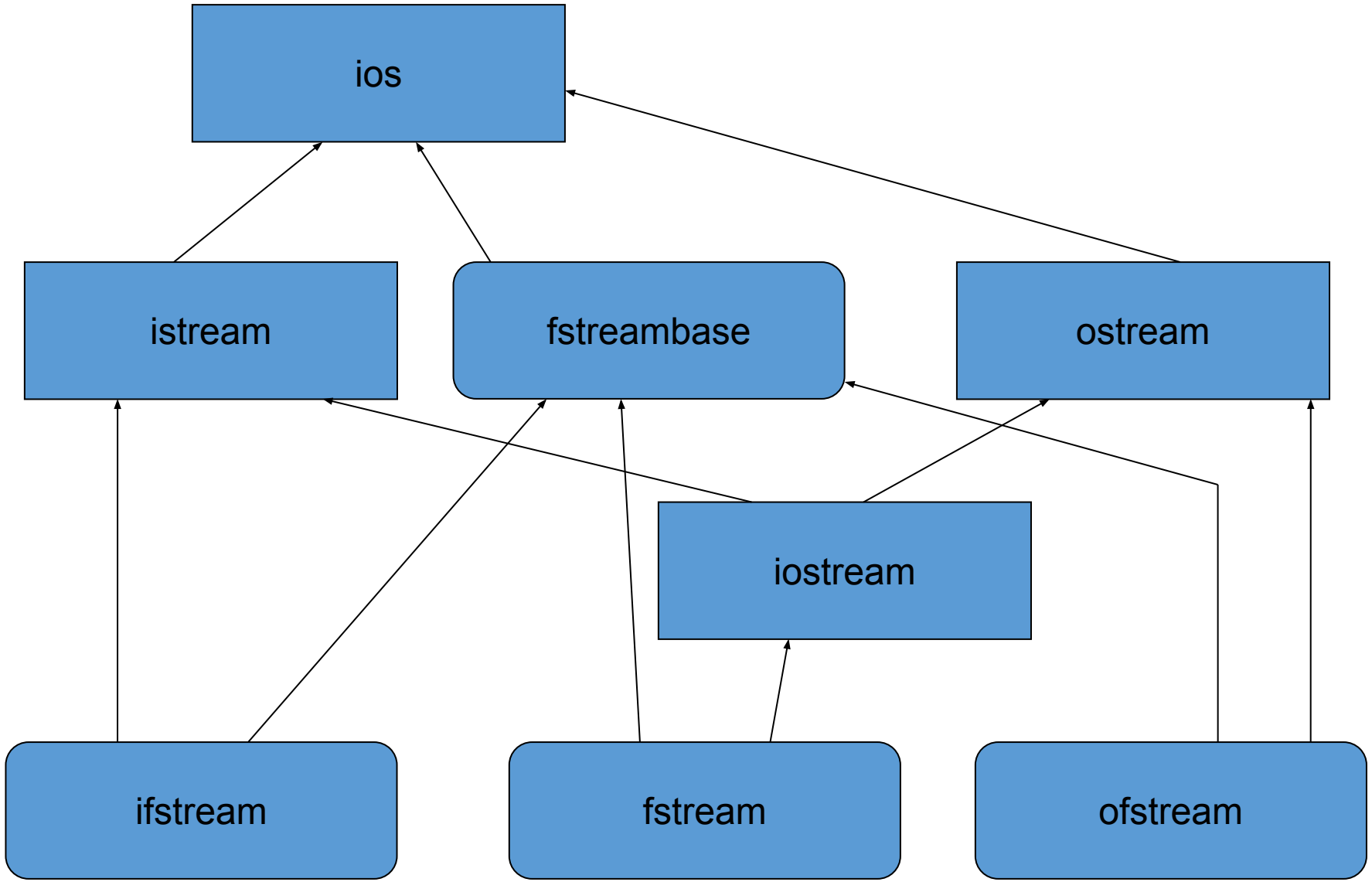
Потоки и файлы

Потоковые классы (упрощенная схема)



Класс ios

- методы
- флаги форматирования *
- флаги статуса ошибки *
- режимы работы с файлами *

* перечисляемые значения, их совокупность представляет битовую маску

Флаги форматирования

skipws – пропуск начальных пробелов при вводе (по умолчанию установлен)

left, right – выравнивание;

internal – заполнение символами между знаком (основанием) и числом

dec, oct, hex – системы счисления

boolalpha – перевод 0 и 1 в true и false

showbase – выводить индикатор системы счисления (0, 0x)

showpoint – показывать десятичную точку

uppercase – показывать буквы в верхнем регистре для 16-ричной системы счисления

showpos – показывать знак '+' положительного числа

scientific, fixed – формат вывода

Методы класса ios

c = fill (), fill (c) – возвращает\устанавливает символ заполнения

p = precision(), precision(p) – возвращает\устанавливает значение точности (число выводимых **знаков**)

w = width(), width(w) – возвращает\устанавливает текущее значение ширины поля (в символах)

unsetf(f), setf(f) - сбрасывает\устанавливает указанный флаг форматирования

setf(flags, field) – очищает поле и устанавливает флаги форматирования

Пример

```
cout.width(10);
cout.fill('_');
cout.unsetf(ios::dec);
cout.setf(ios::hex | ios::right);
cout << 12; //_________c

cout.width(5);
cout.fill('+');
cout.unsetf(ios::hex);
cout.setf(ios::oct | ios::right);
cout << 12 << endl;       //+++14

cout.unsetf(ios::right | ios::oct);
cout.setf(ios::internal | ios::showbase | ios::hex);
cout.fill('_');
cout << 1; //0x__1
```

*Вместо комбинации setf\unsetf можно использовать cout.setf(ios::hex | ios::right,
ios::basefield);

Манипуляторы потоков

- требуют включения заголовочного файла iomanip;
- представляют собой инструкции форматирования, которые вставляются прямо в поток;
- действуют на данные, следующие за ним в потоке (до конца инструкции).

Манипуляторы потоков без аргументов

endl – вывод перехода на новую строку
ends – вывод конца строки (нулевого символа)
dec, oct, hex - вывод в указанной системе счисления
boolalpha – вывод true или false вместо 1 и 0
flush – передача в поток содержимого буфера (только для выходного потока)
ws – пропуск начального символа-разделителя

```
int z = 0xc;
std::cout << z << ", " << std::hex << z;    //12, c
int x = 2;
std::cout << true << ", "
     << std::boolalpha << true << ", "
     << (bool) x << ", "
     << 1 << " " << x << std::endl;

//1, true, true, 1 2
```

Манипуляторы с аргументами

setfill() – установить символ заполнения
setprecision() – установить точность
setiosflags(), resetiosflags() – установить\сбросить флаги форматирования
setw() – устанвить ширину поля для выводимых данных
setbase() – установить базу счисления

Примеры

```
x = 10.0001,
y = 2
new output percision:
x = 10.0001000,
y = 2.0000020
```

```
cin >> a;                  //12
cout << "\nyour input\n"
   << resetiosflags(ios::dec)
   << setiosflags(ios::hex)
   << setw(5) << setfill('+') << a; //0xc++

double x = 10.0001, y = 2.000002;
std::cout << "x = " << x  << ",\n" //only 6 digits
          << "y = " << y << std::endl;
std::cout << std::setprecision(7) << std::fixed
          << "new output percision:\n"
          << "x = " << x << ",\n"
          << "y = " << y << std::endl;   //7 digits
```

Манипулятор setw

```
//for output
cout
 << setw(11) << "Discipline"  << setw(5) << "Mark" << endl
 << setw(11) << "Philosophy"  << setw(5) << "3"    << endl
 << setw(11) << "English"  << setw(5) << "4"    << endl
 << setw(11) << "Geometry" << setw(5) << "5"    << endl;

//for input not greater than MAX symbols
cin >> setw(MAX) >> str;
```

Манипуляторы потоков C++11 (без аргументов)

hexfloat

fixed

scientific

Флаги статуса ошибки

goodbit – ошибок нет

eofbit – достигнут конец файла *

failbit – устанавливается в том случае, если операция завершилась неудачно, но состояние потока данных позволяет продолжить работу. Обычно этот флаг устанавливается при ошибках форматирования в процессе чтения - например, если программа пытается прочитать целое число, а следующий символ является буквой.

badbit – недопустимая операция, указывает на неработоспособность потока данных или потерю данных, например, при установке указателя в файловом потоке данных перед началом файла

*eofbit обычно устанавливается вместе с failbit, поскольку признак конца файла проверяется и обнаруживается при попытке чтения за концом файла. После чтения последнего символа флаг eofbit не устанавливается, он устанавливается вместе с флагом failbit при следующей попытке чтения символа.

Функции флагов ошибки

eof() – true, если eof

fail() – true, если failbit, badbit

bad() – true, если badbit

good() – true, если good

clear(int=0) – устанавливает указанный флаг, без аргумента – снимает все флаги

istream

>> - форматированное извлечение данных из потока

get(c), get(str, max), get(str, max, delim) – извлечение символа или строки из потока (delim остается в потоке)

getline(str, max, delim), getline (str, max) - извлечение строки из потока (delim извлекается из потока)

ignore(max, delim) - извлечение данных из потока

putback(c) – вставка символа во входной поток (буффер потока)

peek(c) – читает один символ из потока без извлечения (возвращает int)

gcount() – возвращает число прочитанных символов последним из вызовов get\read

read(str, size) – извлекает size символов в строку (для файлов)

seekg(pos), seekg(pos, seek_dir) – изменяет указатель чтения (курсор) файла

tellg() – возвращает позицию указателя чтения (курсора) файла

ostream

<< - форматированная вставка данных в поток

put(c) – вставка символа в поток

flush() – очистка буфера и вставка разделителя строк

write(str, size) - вставка size символов из строки в файл

seekp(pos), seekp(pos, seek_dir) - изменяет указатель записи (курсор) файла

tellp() - возвращает позицию указателя записи (курсора) файла

Пример

```
char c, str[10];
cin >> c;//one symbol
cout << "c by cin " << c << endl;
cin.get(c);//one symbol
cout << "c by get " << c << endl;
cin.get(str, 9);//string, not more than 9 symbols before endl
cout << "str by get " << str << endl;
cin.ignore(9, '\n');//ignore 9 symbols or less before endl
cin.get(str, 9, '_');/* string, not more than 9 symbols
                        before  symbol '_' */
```

Пример

```
char str[100];
cin.getline(str, 99);
cout << cin.gcount() << " keys: " << str << endl;
cin.getline(str, 99, '.');
cout.flush();
cout << static_cast<char>(cin.peek());
cin.putback(' ');
cout << str << endl;
cout << cin.get(); //32
cout.put('+');
```

Проверка ввода с использованием битов ошибки

```
int i;
while (true){
  cout << "\ninput integer value\n";
  cin.unsetf(ios::skipws);//разделители
   cin >> i;
   if (cin.good()){
      cin.ignore(10, '\n');
      break;
      }
  if (cin.fail())cout << "fail";
   cin.clear();
   cin.ignore(10,'\n');
   }
cout << endl << i;
```

Пример

```
int i; char str[100];
while (true){
  cout << "\ninput integer value\n";
  cin.unsetf(ios::skipws);
  cin >> str;
  for (int j = 0; j < strlen(str); j++)
    if (str[j] <= '0' || str[j] >= '9'){
        cin.clear(ios::failbit); //устанавливаем
        break; //флаг ошибки вручную
        }
  if (cin.good()){
    cin.ignore(100,'\n');
    break;
    }
  if (cin.fail())cout << "fail";
  cin.clear();
  cin.ignore(100,'\n');
  }
i = atoi(str);
cout << endl << i;
```

Текущий контроль

Выберите один вариант ответа. Что из перечисленного не является манипулятором потока?

1) ends
2) setw
3) oct
4) fix
5) setfill

Текущий контроль

Выберите один вариант ответа. Что из перечисленного не является манипулятором потока?

1) ends
2) setw
3) oct
4) fix
5) setfill

Текущий контроль

Выберите один вариант ответа. Какую функцию выполняет метод putback класса-наследника ios?

1) записывает строку в конец входного потока
2) записывает строку в конец выходного потока
3) записывает символ в конец входного потока
4) записывает символ в конец выходного потока

Текущий контроль

Выберите один вариант ответа. Какую функцию выполняет метод putback класса-наследника ios?

1) записывает строку в конец входного потока
2) записывает строку в конец выходного потока
3) записывает символ в конец входного потока
4) записывает символ в конец выходного потока

Строковые потоки (sstream)

istringstream, производный от istream, читает из строки;

ostringstream, производный от ostream, пишет в строку;

stringstream, производный от iostream, выполняет как чтение, так и запись.

Пример работы со строковыми потоками

```
std::string toString(int x, int y){
std::ostringstream s;
s << "x: " << x << ",y: " << y << std::endl;
return s.str();
}

void fromString(std::string s, int &x, int &y){
std::stringstream ss(s);
std::string sbuf = "";
ss >> sbuf >> x >> sbuf >> y;
}
```

Классы для работы с дисковыми файлами

ifstream – файловый ввод (чтение)

ofstream – файловый вывод (запись)

fstream - двунаправленный файловый поток (поочередная чтение\запись)

Форматированный файловый вывод

```
#include <fstream>
…
int main (){
int i = 99;
float f = 5.375;
char c = '*';
double d = 3.2;
string s = "something";
ofstream tofile("myfile.txt");
tofile << i << ' ' << f << ' ' << c << ' ' << d << ' ' << s;
//99 5.375 * 3.2 something
}
```

Форматированный файловый ввод

```
#include<fstream>
int main (){
char c;
double d;
int i;
float f;
string s;
ifstream fromfile("myfile.txt");
fromfile >> i >> f >> c >> d >> s;
cout << i << ' ' << f << ' ' << c << ' ' << d << ' ' << s;
}
```

Методы open\close

void open(const char *filename, int mode);

void close();

bool is_open();

```
ifstream file1, file2;
string s;
cin >> s;
file1.open("C:\\temp\\1.data", ios::binary);
file2.open(s.c_str());
cout << file1.is_open() << endl;
file1.close();
```

Биты режимов открытия файлов

in - ввод

out - вывод

ate – перенос курсора в конец

app – запись только в конец

trunc – удаление содержимого (создание при отсутствии файла)

binary – двоичный формат чтения\записи

Комбинации битов режима

ios::out | ios::trunc удаляется существующий файл и (или) создается для записи;

ios::in | ios::out | ios::trunc существующий файл удаляется и (или) создается для для чтения и записи;

ios::in | ios::out открывается существующий файл для чтения и записи:;

ios::out | ios::app открывается существующий файл для дозаписи в конец файла.;

ios::in | ios::out | ios::app открывается существующий файл для чтения и дозаписи в конец файла.

Файловый ввод\вывод строк

```
ofstream tofile("myfile.txt");
tofile << "First string\n";
tofile << "Second string\n";
tofile << "Third string\n";
char buff[100];
tofile.close();
ifstream fromfile("myfile.txt");
while (!fromfile.eof()){
  fromfile.getline(buff, 100);
  cout << buff << endl;
  }

/* можно проверять открытие
if (fromfile.good()) или   if (fromfile)…*/

//First string\nSecond string\n Third string\n
```

Посимвольный файловый ввод\вывод

```
string s = "Some string";
ofstream tofile("myfile.txt");
for (int i = 0; i < s.size(); i++)
   tofile.put(s[i]);
char c;
tofile.close();
ifstream fromfile("myfile.txt");
while (fromfile){
  fromfile.get(c);
  cout << c;
  }
}

/*ifstream fromfile("myfile.txt");
cout << fromfile.rdbuf();*/
```

Форматированный файловый вывод структурного объекта

```
struct Student{
 unsigned short Course;
 char FirstName[15];
 char LastName[15];
 char Speciality[10];
};

int main (){
Student Petrov = {1, "Petrov", "Petr", "MOAIS"};
ofstream tofile("myfile.stud");
tofile << Petrov.Course     << endl
     << Petrov.FirstName  << endl
     << Petrov.LastName   << endl
     << Petrov.Speciality << endl;
}
```

Форматированный файловый ввод структурного объекта

```
struct Student{
  unsigned short Course;
  char FirstName[15];
  char LastName[15];
  char Speciality[10];
};

int main (){
Student PetrovsCopy;
ifstream fromfile("myfile.stud");
fromfile >> PetrovsCopy.Course
         >> PetrovsCopy.FirstName
         >> PetrovsCopy.LastName
         >> PetrovsCopy.Speciality;
}
```

Форматированный файловый вывод массива структурированных данных

```
struct Student;

int main (){
int n;
cin >> n;
Student *stud = new Students[n];
//тут будет заполнение массива
ofstream tofile("myfile.stud");
tofile << n << endl; //записать размер массива
for (int i = 0; i < n; i++)
   tofile << stud[i].Course     << endl
          << stud[i].FirstName  << endl
          << stud[i].LastName     << endl
          << stud[i].Speciality << endl;
}
```

Форматированный файловый ввод массива структурированных данных

```
struct Student;


int main (){
int n;
ifstream fromfile("myfile.stud");
fromfile >> n;
Student *stud = new Students[n];
for (int i = 0; i < n; i++)
  fromfile >> stud[i].Course
           >> stud[i].FirstName
           >> stud[i].LastName
           >> stud[i].Speciality;
}
```

reinterpret_cast

Используется для приведения несовместимых типов — ответственность на программисте

```
int i = 68;
//char *pi = (char*)&i;
char *pi = reinterpret_cast<char*>(&i);
cout << *pi << endl; // D
```

К слову о кастомизации: const_cast

```
const float c = 7.1f;
 /*убирает cv спецификаторы, то есть const и volatile*/
float *d = const_cast<float*>(&c);
*d = 8;
//float* e = &c;
//cannot initialize a variable of type
//float* with const float*
std::cout << "c=" << c << ",d=" << *d;
```

Двоичный файловый ввод\вывод массива чисел

```
int imas[100];

//тут должно быть заполнение imas

ofstream tofile("myfile.data", ios::binary);
tofile.write(reinterpret_cast<char*>(imas), 100 *
 sizeof(int));
tofile.close();

//допустим здесь очистка значений imas

ifstream fromfile("myfile.data", ios::binary);
fromfile.read(reinterpret_cast<char*>(imas),   100 *
 sizeof(int));
```

Двоичный файловый ввод\вывод массива структурированных данных

```
struct Student;//should be POD - Plain Old Data
//string is not POD! See vector`s write\read
int main (){
Student Petrov = {1, "Petrov", "Petr", "MOAIS"};
ofstream tofile("myfile.stud", ios::binary);
tofile.write(reinterpret_cast<char*>(&Petrov),
sizeof(Student));
tofile.close();

Student PetrovsCopy;
ifstream fromfile("myfile.stud", ios::binary);
fromfile.read(reinterpret_cast<char*>(&PetrovsCopy),
sizeof(Student));
}
```

Двоичный файловый ввод\вывод массива структурированных данных

```
struct Student;
int main (){
const int startSize = 29;
int varSize;
cin >> varSize;
Student group[startSize];
Student *Group = new Student[varSize];
//тут должно быть заполнение массива group какими-то данными
ofstream tofile("myfile.stud", ios::binary);
tofile.write(reinterpret_cast<char*>(group),
             startSize * sizeof(Student));
tofile.close();
ifstream fromfile("myfile.stud", ios::binary);
fromfile.read(reinterpret_cast<char*>(Group),
              varSize * sizeof(Student));
}
```

Двоичный файловый ввод\вывод vector

```
int main(){
const int n = 5;
vector<int> v{1, 2, 3, 4, 5};
fstream f("1.txt", ios::out | ios::binary);
f.write(reinterpret_cast<char*>(&v[0]), sizeof(v[0]) * v.size());
f.close();
vector<int> v2(v.size());
fstream f2("1.txt", ios::in | ios::binary);
f2.read(reinterpret_cast<char*>(&v2[0]),  sizeof(v2[0]) * v2.size());
for(size_t i = 0; i < v2.size(); i++)
    cout << v2[i] << " ";
}
```

*как правило, в фай

```
f.write(reinterpret_cast<char*>(v.begin()), sizeof(v[0]) * v.size());
```

л записывают сначала количество элементов, затем сами элементы

Двоичный файловый ввод\вывод vector с сохранением информации о размере вектора

```
int main(){
vector<int> v{1, 2, 3, 4, 5};
fstream f("1.txt", ios::out | ios::binary);
size_t vsize = v.size();
f.write(reinterpret_cast<char*>(&vsize), sizeof(vsize));
f.write(reinterpret_cast<char*>(&v[0]), sizeof(v[0]) * vsize);
f.close();
fstream f2("1.txt", ios::in | ios::binary);
size_t v2size;
f2.read(reinterpret_cast<char*>(&v2size), sizeof(v2size));
vector<int> v2(v2size);
//vector<int>v2;
//v2.resize(v2size);
f2.read(reinterpret_cast<char*>(&v2[0]),  sizeof(v2[0]) * v2size);
for(size_t i = 0; i < v2.size(); i++)
    cout << v2[i] << " ";

string s = "abc";
s.write(s.c_str(), s.size());

}
```

С векторами удобно использовать потоковые итераторы (форматированная, не бинарная запись\чтение)

```
vector<int> vo {1, 2, 3, 4, 5};
ofstream ofile("file.dat");
ofstream_iterator<int> osIter(ofile, " ");
//output to file
copy(vo.begin(), vo.end(), osIter);
ofile.close();
vector <int> vi(vo.size());
ifstream ifile("file.dat");
istream_iterator<int> endOfStream;
istream_iterator<int> fileIter(ifile);
//input from file
copy(fileIter, endOfStream, vi.begin());
//output to console
ostream_iterator<int> cosIter(cout, ",");
copy(vi.begin(), vi.end(), cosIter);
```

Текущий контроль

Выберите один или несколько вариантов ответа. Какой из перечисленных классов языка C++ можно использовать для чтения данных из файла в программу?

1) ifstream
2) ofstream
3) fstream
4) iostream
5) stringstream
6) ios / ios_base

Текущий контроль

Выберите один или несколько вариантов ответа. Какой из перечисленных классов языка C++ можно использовать для чтения данных из файла в программу?

1) **ifstream**
2) ofstream
3) **fstream**
4) iostream
5) stringstream
6) ios / ios_base

Текущий контроль

Сколько байт будет занимать файл 1.txt после выполнения программы?

```
#include<fstream>

using namespace std;

int main(){
fstream f("1.txt",ios::out|ios::trunc);
string s = "string";
int i = 12345;
double d = 1.5;
f << i << s << d;
return 0;
}
```

Текущий контроль

Сколько байт будет занимать файл 1.txt после выполнения программы?
Какие данные будут храниться в файле?

```
#include<fstream>

using namespace std;

int main(){
fstream f("1.txt",ios::out|ios::trunc|ios::binary);
string s = "string";
int i = 12345;
double d = 1.5;
f.write(reinterpret_cast<char*>(&s),s.length());
f.write(reinterpret_cast<char*>(&i),sizeof(i));
f.write(reinterpret_cast<char*>(&d),sizeof(d));
return 0;
}
```

Текущий контроль

Какие значения будут храниться в переменных i, f и c после выполнения программы?

```
#include<fstream>
int main ()
{
int i = 99;
float f = 5.375;
char c = '*';
double d = 3.2;
string s = "something";
fstream file("myfile.txt");
file << i << f << c << d << s;
file.close();
file.open("myfile.txt");
file >> i >> f >> c;
return 0;
}
```

Текущий контроль

Введите ответ на вопрос.
В файле 1.txt хранится следующий набор символов:

```
99Masha, 7 Ilya smart, 28Roza
```

Что будет хранится в переменной points после работы следующей программы:

```
ifstream fin("1.txt");
int points;
string student;
while (!fin.eof()){
fin >> points;
getline(fin, student, ',');
}
```

Текущий контроль

Введите ответ на вопрос.
В файле 1.txt хранится следующий набор символов:

```
99Masha, 7 Ilya smart, 28Roza
```

Что будет хранится в переменной points после работы следующей программы:

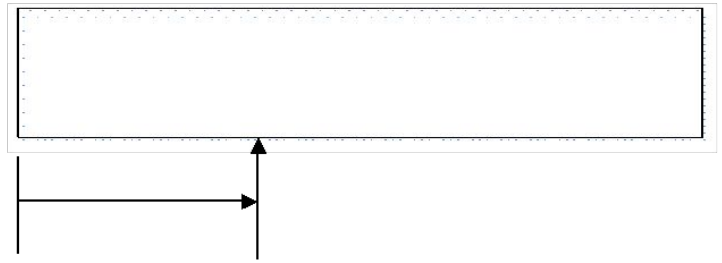
```
ifstream fin("1.txt");
int points;
string student;
while (!fin.eof()){
fin >> points;
getline(fin, student, ',');
}
```

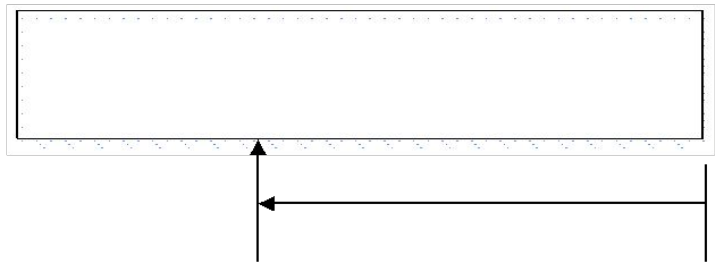
Ответ: 28

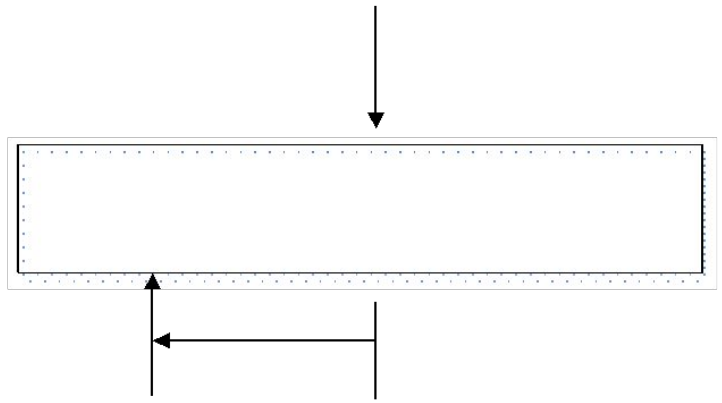
Указатель (курсор) файла

- seekg(int);
- seekg(int, pos);
- tellg();
- seekp(int);
- seekp(int, pos);
- tellp();

- streampos: ios::beg, ios::cur, ios::end







Определение числа структурированных записей в файле

```
struct Student; //POD


int main (){
ifstream fromfile("myfile.stud", ios::binary);
fromfile.seekg(0, ios::end);
int n = fromfile.tellg() / sizeof(Student);
fromfile.seekg(0, ios::beg);
Student *stud = new Student[n];
fromfile.read(reinterpret_cast<char*>(stud), n *
 sizeof(Student));
}
```

Пример записи в конец файла

```
struct Student;

int main (){
Student student;
//ввод данных о студенте
ofstream ofile("myfile.stud", ios::binary | ios::app);
ofile.write(reinterpret_cast<char*>(&student), sizeof(Student));
ofile.close();
ifstream ifile;
ifile.open("myfile.stud", ios::binary);
ifile.read(reinterpret_cast<char*>(&student), sizeof(Student));
while(!ifile.eof()){
  //вывод информации о студенте student
  ifile.read(reinterpret_cast<char*>(&student), sizeof(Student));
  }
}
```

Пример чтения из произвольной позиции в файле

```
struct Student;

int main (){
Student student;
ifstream fromfile;
fromfile.open("myfile.stud", ios::binary | ios::in);
fromfile.seekg(0, ios::end);
int endpos = fromfile.tellg();
int count = endpos / sizeof(Student);
int number;
cin >> number; //ввести номер студента
fromfile.seekg((number - 1) * sizeof(Student));
file.read(reinterpret_cast<char*>(&student), sizeof(Student));
//вывести информацию о студенте student
}
```

streampos

```
streampos pos = f.tellp();
cout << endl << pos << endl;
f.seekg(pos);
while(!f.eof())
cout << (char)f.get();
f << "after";
```

streampos

```
fstream f("1.txt", ios::out | ios::app);
/*with app always append to the file`s end
in other cases makes bias by spaces in count of code value before writing*/
f.seekp(' ');/*code 32, after 32 symbols will be output to file*/
streampos pos = f.tellp();//save output position
f.seekg(pos); //set file input position like pos
while(!f.eof())
cout << (char) f.get(); //show all text after pos
f << "some text"; /*file output "some text" after pos*/
```

* изначально вопрос студента был "почему оно не ищет позицию пробела в файле", отсюда появился этот слайд

eof и seek

```
struct SomeStruct{ /*поля a и b*/};

int main (){
SomeStruct sOb;
ifstream file("C:\\temp\\myfile.st", ios::binary);
while(true){
   file.read(reinterpret_cast<char*>(&sOb), sizeof(sOb));
/*после чтения последней записи необходимо сделать еще 1 считывание,
чтобы установился флаг eof */
   if(file.eof()) break;
   cout << stObj.a << ' ' << stObj.b << endl;
   }
//при необходимости повторного чтения файла
//file.seekg(0);//не работает надо сбросить флаг eof
file.clear(); /*сбрасывает все флаги ошибки, но указатель все еще в
конце файла*/
file.seekg(0); //смещаем указатель файла
file.read(reinterpret_cast<char*>(&sOb), sizeof(sOb));
cout << sOb.a << ' ' << sOb.b; // 1-я запись в файле
```

Чтение и запись в конец файла в одной функции

```
struct SomeStruct{ /*поля a и b*/};

int main (){
SomeStruct sOb;
ifstream file("C:\\temp\\myfile.st", ios::binary);
file.read(reinterpret_cast<char*>(&sOb), sizeof(sOb));
cout << sOb.a << ' ' << sOb.b << endl;

file.close();/*далее обращаемся к тому же файлу, поэтому текущий
поток надо закрыть*/
ofstream file("C:\\temp\\myfile.st", ios::binary | ios::app);
//без app - пересоздает файл

/*file.seekp(sizeof(student)); запись может быть только в конец -
игнорирует seekp*/

file.write(reinterpret_cast<char*>(&sOb), sizeof(sOb));
}
```

Проверка ошибок при файловом вводе\выводе

```
ifstream fromfile;
ofstream tofile;
fromfile.open("myfile1.data", ios::binary);
if (!fromfile)…//обработка ошибки
tofile.open("myfile2.data", ios::binary);
if (!tofile)…//обработка ошибки
```

Аргументы командной строки

```
int main(int argc, char* argv[]) {
cout << "args count = " << argc << endl;
for (int j = 0; j < argc; j++) //output all args
   cout << "arg " << j << " = " << argv[j] << endl;
return 0;
}

/*Например, следующая командная строка:
C:\temp>e.exe arg1 12.6 1234567 lastarg
Выведет на экран:
args count = 4
arg 0 = arg1
arg 1 = 12.6
arg 2 = 1234567
arg 3 = lastarg
*/
```

Перенаправление потоков ввода\вывода посредством командной строки

```
int main(){
int a;
cin >> a;
cout << a + 1;
}
```

Записывает выходные данные main.exe в конец файла 2.txt

```
main.exe >> c:\temp\2.txt
```

Перезаписывает файла 2.txt выходными данными main.exe

```
main.exe > c:\temp\2.txt
```

Подает на вход main.exe данные из 1.txt и записывает выходные данные в 2.txt

```
main.exe < 1.txt > c:\temp\2.txt
```

Работа с файлами Функции в языке C (stdio.h)

```
FILE *filePointer;
FILE *fopen(const char *fileName, const char *openMode);
//modes: a, r, w, a+, r+, rb, wb…
int fclose(FILE *filePointer);
int putc(int symbol, FILE *filePointer);
int getc(FILE * filePointer);
int feof(FILE * filePointer);
int fputs(const char *str, FILE * filePointer);
char *fgets(char *str, int len, FILE * filePointer);
size_t fwrite(const void *writingData, size_t sizeOfData, size_t dataCount,
FILE *filePointer);
size_t fread(void *readingData, size_t sizeOfData, size_t dataCount, FILE
*filePointer);
long ftell(FILE *filePointer);
int fseek(FILE *filePointer, long int bytesCount, int startPoint);
//startPoint: SEEK_SET, SEEK_CUR, SEEK_END
…
```

Примеры работы с файлами в языке С

```
int n;
scanf("%d", &n);
float *a = (float*)malloc(n * sizeof(float));
for(int i = 0; i < n; i++)
    a[i] = (i + 1) / 2.0;
char myFileName[] = "1.float";
FILE *out = fopen(myFileName, "w");
fwrite((void*) &n, sizeof(n), 1, out);
fwrite((void*) a, n * sizeof(float), n, out);
//fwrite((void*) a, n * sizeof(a[0]), n, out);
fclose(out);
FILE *in = fopen(myFileName, "r");
int m = 0;
fread((void*) &m, sizeof(m), 1, in);
float *b = (float*)malloc(m * sizeof(float));
fread((void*)b, m * sizeof(float), m, in);
```

Системный ввод\вывод (io.h)

chsize - изменяет размер файла
filelength - контролирует длину файла
locking - защищает области файла от несанкционированного доступа.
mktemp - создает уникальное имя файла
remove - удаляет файл
rename - переименовывает файл
setmode - устанавливает режим обработки файла
stat - получает по имени файла информацию о статусе файла