

**Объектно-ориентированное  
программирование  
с использованием языка**

**C++**



# Урок №11

Объектно-ориентированное программирование с использованием языка C++

## Содержание

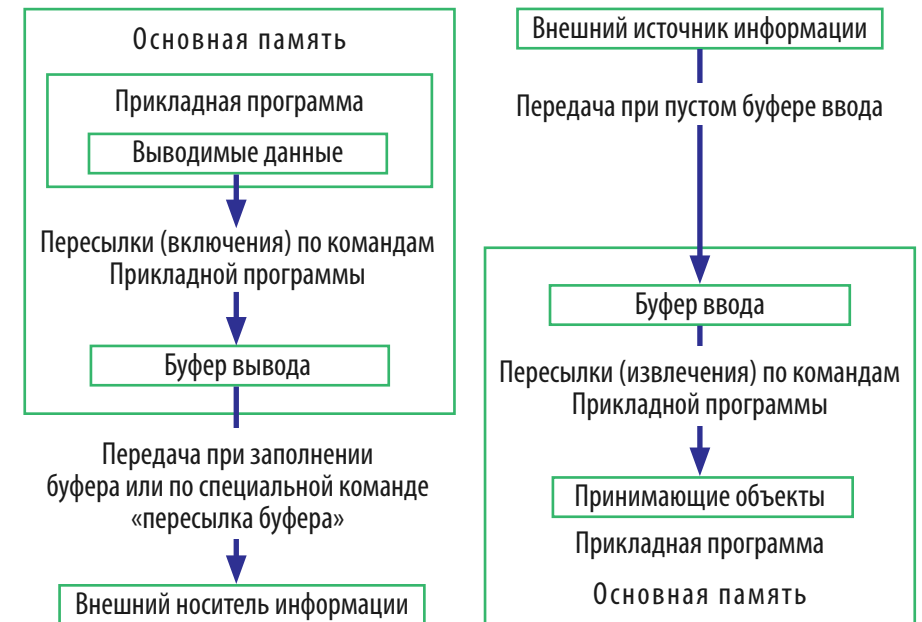
Понятие потока .....	3
Ввод-вывод в языке C++ .....	6
Основные утверждения, связанные с библиотекой <code>iostream</code> .....	7
Файловый ввод-вывод с применением потоков .....	9
Функции для работы с файлами .....	11
Практический пример использования файлов в стиле C++ .....	16
Практический пример. Ввод/вывод массива в/из файл(-а) .....	19
Практический пример записи объекта класса в файл .....	23
Домашнее задание .....	30

## Понятие потока

Сегодня мы снова будем говорить о файлах, но рассмотрим их совсем с другой позиции. Однако, для начала, необходимо нам познакомиться с некоторыми новыми понятиями и схемами. Такими как поток и принцип его работы.

**Поток** — последовательность байтов, не зависящая от устройств, с которыми ведется обмен данными.

**Примечание:** Обмен данными с потоком часто сопровождается участием вспомогательного участка памяти — **буфера потока**.



Процессы, изображенные на рисунках, протекают следующим образом:

- Перед тем, как данные, выводимые программой, будут переданы к внешнему устройству, они помещаются в буфер потока.
- При вводе данных они вначале помещаются в буфер и только затем передаются в область памяти выполняемой программы.
- Использование буфера повышает скорость передачи данных, так как настоящие пересылки осуществляются только тогда, когда буфер уже заполнен (при выводе) или пуст (при вводе).

В программах классификация потоков представляет собою следующее:

- *Входные потоки* — потоки, из которых читается информация.
- *Выходные потоки* — потоки, в которые вводятся данные.
- *Двунаправленные потоки* — потоки, допускающие как чтение, так и запись.

Отметим, что работу, связанную с заполнением и очисткой буферов ввода-вывода, операционная система практически всегда берет на себя и выполняет без участия программиста. Очень важно, что никакой связи значений байтов, содержащихся в потоке с кодами какого-либо алфавита не предусматривается. Задача программиста при вводе-выводе с помощью потоков — установить связь

между участвующими в обмене объектами и последовательностью байтов потока, в которой отсутствуют всякие сведения о типах передаваемой информации. Именно этим мы и попытаемся сегодня заняться.

# Ввод-вывод в языке C++

Напомним, что с самого начала изучения языка программирования C, и его логического продолжения C++, мы подключали к нашим программам библиотеку `iostream`. На самом деле предназначение этой библиотеки встроить в программу часть библиотеки ввода-вывода, построенной на основе механизма классов.

Описанные в этом файле средства ввода-вывода предоставляют нам орудие для извлечения данных из потоков и для включения данных в потоки.

**Примечание:** Расшифруем название: *iostream.h*: *stream* — поток, "i" — сокр. input — ввод, "o" — сокр. output — вывод.

В библиотеке `iostream` есть четыре предопределенных объекта-потока, ассоциированных со стандартным вводом и выводом. Рассмотрим их:

1. **cin** — объект класса `istream`, связанный со стандартным буферизированным входным потоком (обычно с клавиатурой).
2. **cout** — объект класса `ostream`, связанный со стандартным буферизированным выходным потоком (обычно с дисплеем).
3. **cerr** — объект класса `ostream`, связанный со стандартным небуферизированным выходным потоком (обычно

с дисплеем), в который направляются сообщения об ошибках.

4. **clog** — то же, что **cerr**, но буферизирован.

При подключении к программе файла `iostream` происходит формирование объектов `cin`, `cout`, `cerr` и `clog`, т.е. создаются соответствующие стандартные потоки, и программисту становятся доступными связанные с ними средства ввода-вывода.

## Основные утверждения, связанные с библиотекой `iostream`

- Библиотека потоков C++ предусматривает два основных класса для ввода и вывода: `istream` и `ostream`.
- Класс `ostream` использует для вывода переопределенную операцию левого сдвига (`<<`), которую называют операцией помещения в поток.
- Класс `istream` использует для ввода переопределенную операцию правого сдвига (`>>`), которую называют операцией извлечения из потока.
- Операции помещения и извлечения допускают сцепленные вызовы, т.к. возвращают значение ссылки на поток.
- Классы `istream` и `ostream` перегружают операции помещения и извлечения для всех встроенных типов данных. Такая перегрузка позволяет использовать единый синтаксис для ввода и вывода символов, строк, целых и вещественных чисел.

- Операции помещения/извлечения можно легко распространить на пользовательские типы данных. Для этого надо определить две функции со следующими заголовками:

```
istream& operator >> (istream&, имя_типа&);  
ostream& operator << (ostream&, имя_типа&);
```

**Примечание:** Определяя операции помещения/извлечения для классов, их обычно делают дружественными, чтобы обеспечить доступ к закрытым элементам данных класса.

Только что мы разобрали потоки, связанные с выводом на дисплей. Пора переходить к файлам.

## Файловый ввод-вывод с применением потоков

Итак. Вы, наверняка, согласитесь, что файл — это именованная цепочка байтов, у которой есть начало и конец. В C++ существует библиотека под названием `fstream`. С её помощью, можно осуществить следующие функции:

- Создание файла.
- Создание потока.
- Открытие файла.
- «Присоединение» файла к потоку.
- Обмен с файлом с помощью потока.
- «Отсоединение» потока от файла.
- Закрытие файла.
- Уничтожение файла.

**Примечание:** Для корректной работы библиотеки `fstream` необходимо использовать `using namespace std;`

Библиотека `fstream`, как и библиотека `iostream` содержит три класса, предназначенных для ввода и вывода данных в файлы:

- **`ofstream`** — для вывода (записи) данных в файл.
- **`ifstream`** — для ввода (чтения) данных из файла.
- **`fstream`** — для чтения и записи данных.

Для каждого из этих трех классов предусмотрено четыре конструктора. Рассмотрим их:

```

fstream() - создает поток, не открывая файла;
fstream(
    const char* name,          //имя файла
    int omode,                 //режим открытия
    int = filebuf::openprot    //защита файла
) - создает поток, открывает файл и связывает его
    с потоком;

fstream(
    int f                      // дескриптор файла
) - создает поток и связывает его с уже открытым
    файлом

fstream(
    int f,                     //дескриптор файла
    char *buf,                 //буфер
    int len                     //размер буфера
) - то же, что предыдущий конструктор,
    но потоку назначается буфер.

```

Среди вышеперечисленных параметров можно выделить параметр `omode`. Это набор флагов для открытия файла:

```

enum _Openmode {
    in      = 0x01, //открыть только для чтения
    out     = 0x02, //открыть только для записи
    ate     = 0x04, //установить указатель на
                  //конец файла
    app     = 0x08, //дописывать данные в конец файла
    trunc   = 0x10, //усечь файл до нулевой длины
    _Nocreate = 0x40, //если файл не существует,
                  //ошибка открытия
    _Noreplace = 0x80, //если файл уже существует,
                  //ошибка открытия
    binary  = 0x20 //открыть файл для двоичного
                  //обмена
};

```

**Примечание:** Для определения флага к нему необходимо добавить `ios::`. Например, `ios::in`.

## Функции для работы с файлами

```

void open(const char *fileName, int mode = знач_по_ум,
          int protection = знач_по_ум);

```

**«Присоединяет» файловый поток к конкретному файлу.**

**fileName** — имя уже существующего или создаваемого заново файла. Это строка, которая задает полное или сокращенное имя файла в формате, определенном операционной системой.

**mode** — режим открытия.

**protection** — защита файла.

Функция вызывается через объект любого из трех потоковых классов и записывает в него нуль в случае ошибки.

```

int close();

```

Функция очищает буфер потока, отсоединяет поток от файла и закрывает файл.

**Примечание:** Эта функция вызывается автоматически при завершении программы.

```

istream& istream::read(unsigned char *buf, int len);
istream& istream::read(signed char *buf, int len);

```

Производит чтение блока символов.

**len** — максимальное число символов, которые должны быть извлечены из потока в буфер **buf**.

```
ostream&amp;
    ostream::write(const unsigned char *buf, int n);
ostream&amp;
    ostream::write(const signed char *buf, int n);
```

Производит запись блока символов.

**n** — число символов, считая ноль-символ, которые должны быть помещены в поток из буфера **buf**.

```
int istream::get();
istream& istream::get(unsigned char&);
istream& istream::get(signed char&);
```

Извлекает один символ из потока.

```
ostream& ostream::put(char);
```

Помещает один символ в поток.

```
istream& istream::get(unsigned char *buf, int n,
    char c = '\n');
istream& istream::get(signed char *buf, int n,
    char c = '\n');
```

Извлекает строку из потока.

Во всех вышеописанных функциях символы извлекаются и помещаются в буфер пока не будет найден символ-ограничитель, или не будет прочитано **n** символов, или не встретится конец файла. Ограничитель из потока не извлекается и в буфер не помещается.

```
istream& istream::getline(unsigned char *buf, int n,
    char c = '\n');
istream& istream::getline(signed char *buf, int n,
    char c = '\n');
```

Данная функция выполняет то же, что и **get**, но ограничитель извлекается из потока (в буфер не заносится).

```
istream& istream::ignore(int n = 1, int d = EOF);
```

Данная функция извлекает символы из потока, пока не встретится ограничитель **d** или не извлечет **n** символов.

```
int istream::gcount();
```

Данная функция возвращает число символов, извлеченных последней функцией бесформатного ввода.

```
int istream::peek();
```

Данная функция позволяет "взглянуть" на очередной символ входного потока — возвращает код следующего символа (или EOF, если поток пуст), но оставляет этот символ в потоке. При необходимости этот символ можно извлечь из потока с помощью других средств библиотеки.

```
istream& istream::putback(char cc);
```

Данная функция не извлекает ничего из потока, а помещает в него символ **cc**, который становится текущим и будет следующим извлекаемым из потока символом.

```
istream& istream::seekg(long pos);
```

Данная функция устанавливает позицию чтения из потока в положение, определяемое значением параметра.

```
istream& istream::seekg(long off, ios::seek_dir dir);
```

Данная функция выполняет перемещение позиции чтения вдоль потока в направлении, определенном параметром `dir`, который принимает значение из перечисления `enum seek_dir {beg, cur, end};`. Относительная величина перемещения (в байтах) определяется значением параметра `long off`. Если направление определено как `beg`, то смещение от начала потока; `cur` — от текущей позиции; `end` — от конца потока.

```
ostream& ostream::seekp(long pos);
```

Данная функция устанавливает абсолютную позицию записи в поток.

```
ostream& ostream::seekp(long off, ios::seek_dir dir);
```

Данная функция аналогична функции `seekg()`, но принадлежит классу `ostream` и выполняет относительное перемещение записи в поток.

```
long istream::tellg();
```

Данная функция определяет текущую позицию чтения из потока.

```
long ostream::tellp();
```

Данная функция определяет текущую позицию записи в поток.

Ну, что же теории, пожалуй достаточно — пора переходить к практике. В следующих разделах урока мы рассмотрим несколько примеров.



# Практический пример использования файлов в стиле C++

Ну что же, пора переходить к практике. Сейчас, мы с вами рассмотрим пример, который, используя средства языка C++, реализует возможность просмотра файла в шестнадцатеричном виде.

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string.h>
#include <conio.h>
using namespace std;

//Максимальная длина пути к файлу
#define MAX_PATH 260
//Количество столбцов на экране
#define NUM_COLS 18
//Количество строк на экране
#define NUM_ROWS 24

void main()
{
    char path[MAX_PATH];
    //Запрашиваем путь к файлу
    cout << "Input path to file: ";
    cin.getline(path, MAX_PATH);

    int counter = 0, i = 0, j = 0;
    char text[NUM_COLS];
```

```
//Открытие файла в двоичном режиме
ifstream input(path, ios::in | ios::binary);
if (!input)
{
    cout << "Cannot open file for display!" << endl;
    return;
}

//Режим отображения в верхнем регистре для
//шестнадцатеричного вывода
cout.setf(ios::uppercase);

//Пока не конец файла, читаем из него данные
//и производим форматированный вывод на экран
while (!input.eof())
{
    //Посимвольное чтение
    for (i = 0; (i < NUM_COLS && !input.eof()); i++)
        input.get(text[i]);

    if (i < NUM_COLS)
        i--;

    for (j = 0; j < i; j++)
        if ((unsigned)text[j] < 0x10)

            //Количество символов для отображения числа
            //меньше двух?
            cout << setw(2) << 0 << hex << (unsigned) text[j];
        else
            cout << setw(3) << hex << (unsigned) text[j];
    //Выравнивание для незавершенной строки
    for (; j < NUM_COLS; j++)
        cout << "      ";
    cout << "\t";
```

## Практический пример. Ввод/вывод массива в/из файл(-а)

Пример программы, которая осуществляет запись массива в файл или чтение его из файла. Информация хранится в следующем виде:

- Первая строка — размерность массива (через пробел количество строк и столбцов).
- Все последующие строки — элементы массива.

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

void main()
{
    char Answer;
    const MessageCount = 8;
    int i, j;

    //Подсказки
    enum {CHOICE = 3, INPUT_FILENAME,
          INPUT_DIMENSIONS, INPUT_ELEMENTS,
          FILE_ERROR};

    //Сообщения
    char Msg[MessageCount][50] =
    {
```

```
for (j = 0; j < i; j++)
    //Символ не является управляющим?
    if(text[j] > 31 && text[j] <= 255)
        cout << text[j];
    else
        cout << ".";
cout << "\n";

//Если экран уже заполнен,
//производим остановку
if (++counter == NUM_ROWS)
{
    counter = 0;
    cout << "Press any key to continue..." <<
        flush;

    //Ожидаем нажатия любой клавиши
    getch();
    cout << "\n";
}
}
//Закрываем файл
input.close();
}
```

```

    "1. Вывести данные из текстового файла\n",
    "2. Записать данные в текстовый файл\n",
    "3. Выход из программы\n",
    "\nВаш выбор: ",
    "Введите имя обрабатываемого файла: ",
    "Введите размерности матрицы:\n",
    "Введите элементы матрицы:\n",
    "Невозможно открыть файл\n"
};

//Русификация сообщений и вывод меню на экран

for(i = 0; i < MessageCount; i++)
    CharToOem(Msg[i], Msg[i]);

do
{
    for(int i = 0; i < 4; i++)
        cout << Msg[i];
    cin >> Answer;
} while (Answer < '1' || Answer > '3');

if(Answer == '3')
    return;

//Переменная для имени файла
char FileName[80];

//Размерности матрицы
int M, N;

int num;
cout << "\n" << Msg[INPUT_FILENAME];
cin >> FileName;

//Если выбран первый пункт меню,
//то выводим данные из текстового файла на экран

```

```

if(Answer == '1')
{
    //Если файл с указанным именем не существует,
    //выводим сообщение об ошибке
    ifstream inF(FileName, ios::in | ios::_Nocreate);
    if (!inF)
    {
        cout << endl << Msg[FILE_ERROR];
        return;
    }
    //Считываем размерность массива
    inF >> M;
    inF >> N;
    //Считываем элементы массива из файла и
    //выводим их сразу на экран
    for (i = 0; i < M; i++)
    {
        for (j = 0; j < N; j++)
        {
            inF >> num;
            cout << setw(6) << num;
        }
        cout << endl;
    }
    inF.close();
}

//Если выбран второй пункт меню, то запрашиваем
//у пользователя данные и выводим их в текстовый
//файл
else
{
    //Открываем файл для записи.
    //Если файл с указанным именем не существует,
    //то программа создает его
    ofstream outF(FileName, ios::out);
    if (!outF)
    {

```

## Практический пример записи объекта класса в файл

А, теперь, давайте попробуем использовать изученные сегодня средства для записи содержимого полей объекта класса в файл. Наша программа будет осуществлять до-запись, без затирания предыдущих значений. Отметим, что в данном примере файл должен уже присутствовать на диске, поэтому не забудьте его создать.

```

        cout << "\n" << Msg[FILE_ERROR];
        return;
    }
    //Запрашиваем размерность матрицы и
    //записываем данные в файл
    cout << Msg[INPUT_DIMENSIONS];
    cout << "M: ";
    cin >> M;
    cout << "N: ";
    cin >> N;

    outF << M << ' ' << N << "\n";

    cout << Msg[INPUT_ELEMENTS];
    //Запрашиваем элементы массива и записываем
    //их в файл
    for (i = 0; i < M; i++)
    {
        for(j = 0; j < N; j++)
        {
            cout << "A[" << i << "][" << j << "] = ";
            cin >> num;
            outF << num << " ";
        }
        outF << "\n";
    }
    outF.close();
}

```

```

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

void RussianMessage(char *message) {
    char rmessage[256];
    CharToOem(message, rmessage);
    cout<<rmessage;
}

int RussianMenu() {
    RussianMessage("\nВведите 1 для добавления
                    новой структуры в файл\n");
    RussianMessage("Введите 2 для показа всех
                    структур из файла\n");
    RussianMessage("Введите 3 для выхода\n");
    int choice;
    cin>>choice;
}

```

```

        return choice;
    }

    class Man{
        //переменная для возраста
        int age;
        //переменная для имени
        char *name;
        //переменная для фамилии
        char *surname;
    public:
        //конструктор с параметрами
        Man(char *n,char *s,int a);
        //конструктор
        Man();
        //деструктор
        ~Man();
    public:
        //функция ввода данных
        void Put();
        //функция показа данных
        void Show();
        //функция сохранения в файл
        void SaveToFile();
        //функция показа содержимого файла
        static void ShowFromFile();
    };
    //конструктор
    Man::Man() {
        name=0;
        surname=0;
        age=0;
    }
    //конструктор с параметрами
    Man::Man(char *n,char *s,int a) {
        name=new char[strlen(n)+1];
        if(!name){

```

```

        RussianMessage("Ошибка при выделении памяти !!!");
        exit(1);
    }
    strcpy(name,n);
    surname=new char[strlen(s)+1];
    if(!surname){
        RussianMessage("Ошибка при выделении памяти !!!");
        exit(1);
    }
    strcpy(surname,s);

    age=a;
}

//деструктор
Man::~~Man() {
    delete[] name;
    delete[] surname;
}

//функция ввода данных
void Man::Put() {
    char temp[1024];
    RussianMessage("\nВведите имя:\n");
    cin>>temp;

    if(name)
        delete[] name;

    name=new char[strlen(temp)+1];
    if(!name){
        RussianMessage("Ошибка при выделении памяти !!!");
        exit(1);
    }
    strcpy(name,temp);
    RussianMessage("\nВведите фамилию:\n");
    cin>>temp;

```

```

    if(surname)
        delete[] surname;

    surname=new char[strlen(temp)+1];
    if(!surname){
        RussianMessage("Ошибка при выделении памяти !!!");
        exit(1);
    }
    strcpy(surname,temp);

    RussianMessage("\nВведите возраст\n");
    cin>>age;
}

//функция показа данных
void Man::Show(){
    RussianMessage("\nИмя:\n");
    cout<<name;
    RussianMessage("\nФамилия:\n");
    cout<<surname;
    RussianMessage("\nВозраст:\n");
    cout<<age<<"\n";
}

//функция сохранения в файл
void Man::SaveToFile(){
    int size;
    fstream f("men.txt",ios::out|ios::binary|ios::app);
    if(!f){
        RussianMessage("Файл не открылся для записи !!!");
        exit(1);
    }
    //Записываем возраст
    f.write((char*)&age,sizeof(age));
    size=strlen(name);
    //Записываем количество символов в имени
    f.write((char*)&size,sizeof(int));

```

```

    //Записываем имя
    f.write((char*)name,size*sizeof(char));
    size=strlen(surname);
    //Записываем количество символов в фамилии
    f.write((char*)&size,sizeof(int));
    //Записываем фамилию
    f.write((char*)surname,size*sizeof(char));
    f.close();
}

//функция показа содержимого файла
void Man::ShowFromFile(){
    fstream f("men.txt",ios::in|ios::binary);
    if(!f){
        RussianMessage("Файл не открылся для
                        чтения !!!");
        exit(1);
    }
    char *n,*s;
    int a;
    int temp;
    //В цикле зачитываем содержимое файла
    while (f.read((char*)&a,sizeof(int))){

        RussianMessage("\nИмя:\n");

        f.read((char*)&temp,sizeof(int));
        n=new char[temp+1];
        if(!n){
            RussianMessage("Ошибка при выделении
                            памяти !!!");
            exit(1);
        }
        f.read((char*)n,temp*sizeof(char));
        n[temp]='\0';
        cout<<n;
    }
}

```

```

        RussianMessage("\nФамилия:\n");
        f.read((char*)&temp,sizeof(int));
        s=new
        char[temp+1];
        if(!s){
            RussianMessage("Ошибка при выделении
                            памяти !!!");

            exit(1);
        }
        f.read((char*)s,temp*sizeof(char));
        s[temp]='\0';
        cout<<s;

        RussianMessage("\nВозраст:\n");
        cout<<a<<"\n";
        delete []n;
        delete []s;
    }
}

void main(){
    Man *a;
    //Основной цикл программы
    do{
        switch(RussianMenu()){
            case 1: //Добавление записи
                a=new Man;
                a->Put();
                a->SaveToFile();
                delete a;
                break;
            case 2: //Показ всех записей
                Man::ShowFromFile();
                break;
            case 3: //Прощание с пользователем
                RussianMessage("До Свидания\n");
                return;
        }
    }
}

```

```

        default: //Неправильный ввод
            RussianMessage("Неверный Ввод\n");
        }

    }while(1);
}

```

# Домашнее задание

---

## 1. Создать класс СПРАВОЧНИК со следующими полями:

- а) Название фирмы;
- б) Владелец;
- в) Телефон;
- г) Адрес;
- д) Род деятельности.

## 2. Реализовать следующие возможности:

- Поиск по названию;
- Поиск по владельцу;
- Поиск по номеру телефона;
- Поиск по роду деятельности;
- Показ всех записей и добавление.

Вся информация, естественно, хранится в файле с возможностью дозаписи.