

Практическое занятие № 7 «Работа с файлами в С++»

Учебные цели:

- получение умений и навыков работы с файлами в С++.

Воспитательные цели:

- формировать диалектико-материалистическое мировоззрение;
- формировать навыки самостоятельности и дисциплинированности;
- стимулировать активную познавательную деятельность обучающихся, способствовать формированию у них творческого мышления.

Категория слушателей: 2,3 курс.

Время: 90 мин.

Место проведения: компьютерный класс.

Материально-техническое обеспечение:

1) персональный компьютер *IBM PC* с операционной системой Windows XP; 2) среда разработки приложений *Visual C++ .NET*.

ПЛАН ПРАКТИЧЕСКОГО ЗАНЯТИЯ

Учебные вопросы	Время, мин
Вступительная часть	5
1. Файловый ввод вывод в С++	15
2. Функции состояния	10
3. Выполнение индивидуального задания	55
Заключительная часть	5

Элементы теории

Файловый ввод/вывод. В С++ для работы с файлами обычно используют классы потокового ввода/вывода, определенные в заголовочном файле *fstream*. Эти классы являются наследниками классов *istream* и *ostream*, а значит, наследуют и все их методы. Однако, учитывая специфику файлов, имеются и специальные методы, например методы произвольного доступа к файлу.

- *ifstream* – класс входных файловых потоков;
- *ofstream* – класс выходных файловых потоков;
- *fstream* – класс двунаправленных файловых потоков.

Открытие файла возможно при помощи конструктора файлового потока. Если же используется конструктор без параметров, то создается экземпляр файлового потока, а связь с файлом будет установлена при помощи метода `open ()`. Закрываются потоки методом `close ()`, но можно положиться на деструктор потоковой переменной, который вызовет его автоматически. Для наглядности приведём все методы для открытия и закрытия файла в табл. 1.

Чтобы приступить к записи в файл, необходимо сначала создать объект `ofstream`, а затем связать его с определенным файлом на диске. Использование объекта `ofstream` требует включения в программу файла заголовка `fstream.h`.

Таблица 1

Методы для открытия и закрытия файла

Метод	Действие
<code>open (имя)</code>	Открытие файла потока по умолчанию
<code>open(имя, флаги)</code>	Открытие файла потока с установленными
<code>close ()</code>	Закрытие файлового потока
<code>is_open ()</code>	Проверка открытия файла

Примечание. Поскольку файл заголовка `fstream.h` содержит файл `iostream.h`, нет необходимости подключать его явно.

Функции состояния. Объекты `iostream` содержат флаги, отражающие состояние ввода и вывода, чтение каждого из этих флагов можно проверить с помощью функций, возвращающих значение *true* или *false*: `eof ()`, `bad ()`, `fail ()` и `good ()`. Функция `eof ()` возвращает значение *True*, если объект `iostream` встретил конец файла (EOF – end of file – конец файла). Функция `bad()` возвращает значение *true* при попытке выполнить неверную операцию. Функция `fail ()` возвращает значение *true* каждый раз, когда это значение возвращает функция `bad ()`, а также в тех случаях, когда операция невыполнима в данных условиях. Наконец, функция `good()` возвращает значение *true*, когда идет хорошо, т.е. все остальные функции возвращают значение *false*.

Как открыть файл для ввода и вывода? Чтобы открыть для вывода (записи) файл `test.dat` с помощью объекта `ofstream`, необходимо создать экземпляр объекта класса `ofstream` и передать ему имя файла в качестве параметра:

```
ofstream fout ("test.dat");
```

Чтобы открыть этот файл для ввода (чтения), применяется тот же синтаксис, за исключением указания объекта класса `ifstream`:

```
ifstream fin ("test.dat ");
```

Обратите внимание, что **fout** и **fin** не более чем имена объектов; здесь объект **fout** использовался для вывода (записи) в файл подобно тому, как объект **cout** используется для вывода на экран; объект **fin** аналогичен объекту **cin**.

Очень важным методом, используемым в файловых потоках, является функция-член **close ()**. Каждый раз открывая файл для чтения или записи (или того и другого), создается соответствующий объект файлового потока. По завершении работы файл необходимо закрыть с помощью функции **close ()**, чтобы впоследствии не повредить его и записанные в нем данные.

После того как объекты потока будут ассоциированы с файлами, они используются наравне с другими объектами потока ввода и вывода. Листинг П.1.1 демонстрирует этот подход.

Изменение поведения объекта ofstream. Обычно объект класса **ofstream**, открывая файл для записи, создает новый файл, если таковой не существует, или усекает его длину до нуля, если файл с этим именем уже существует (то есть удаляет всё его содержимое). Изменить стандартное поведение объекта **ofstream** можно с помощью второго аргумента конструктора заданного явно.

Допустимыми аргументами являются:

- **ios :: app** – добавляет данные в конец файла, не усекая (удаляя) прежнее его содержимое;
- **ios :: ate** – осуществляет переход в конец файла, но запись допускает в любом месте файла;
- **ios :: trunc** – усекает существующий файл полностью (задано по умолчанию), т.е. удаляет старое содержимое файла;
- **ios :: nocreate** – открывает существующий файл;
- **ios :: noreplace** – открывает несуществующий файл;
- **ios :: binary** – открывает файл в двоичном режиме.

App – это сокращение от **append** – добавить в конец, **ate** – от **at end** – перейти в конец, **trunc** – от **truncate** – усечение, **nocreate** – не создавать, а **noreplace** не замещать. В листинге П.1.2 будет открыт файл, созданный в листинге П 1.1, чтобы осуществить дозапись данных в конец файла.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОТРАБОТКЕ УЧЕБНЫХ ВОПРОСОВ

Практическое задание

Задание 1.

1) Напишите и отладьте программу, которая выполнит следующие действия:

- создать на диске M:\ в папке с Вашей фамилией файл test.txt;
- записать в файл test.txt две строки 1) Ваши фамилия, имя и отчество;

2) место учебы (ввод данных происходит с клавиатуры);

- вывести содержимое файла test.txt на экран.

2) Напишите и отладьте программу, которая выполнит следующие действия:

• добавить в созданный ранее файл test.txt информацию о Вашей дате рождения и месте жительства;

- вывести содержимое файла test.txt на экран.

Схему алгоритма решения, текст и результаты работы программы запишите в отчет по работе.

Задание 2. Напишите программу, которая считывает из текстового файла in_massiv.txt элементы массива A (размерность массива не более 50 элементов) и обрабатывает их в соответствии с вашим вариантом. Полученные результаты записать в новый текстовый файл out_massiv.txt.

Варианты задания 2.

1. Исключить из массива $A_1..A_N$ первый отрицательный элемент.
2. Исключить из массива $A_1..A_N$ первый четный элемент, следующий за максимальным.
3. Дан массив целых чисел $A_1..A_N$. Выяснить, какая из трех ситуаций имеет место: все числа $A_1..A_N$ равны нулю, в последовательности $A_1..A_N$ первое ненулевое число - положительное, первое ненулевое число - отрицательное. В зависимости от ситуации первым элементом массива записать 1, 2 или 3 соответственно, сдвинув все элементы массива вправо.
4. Дан массив целых чисел $A_1..A_N$. Заменить все отрицательные числа квадратами их значений, а положительные числа - минимальным элементом массива.
5. Даны целые числа $A_1..A_N$. Определить количество целых чисел, входящих в последовательность $A_1..A_N$ по одному разу. Данное число записать в конец массива.
6. Дан массив действительных чисел $A_1..A_N$. Найти максимальный элемент среди отрицательных элементов и поменять его местами с минимальным положительным.

7. Перенести в начало одномерного массива второй нулевой элемент.
8. В одномерном массиве перенести в конец минимальный элемент.
9. Перенести в хвост одномерного массива все отрицательные элементы.
10. Перенести в начало одномерного массива все нечетные элементы.

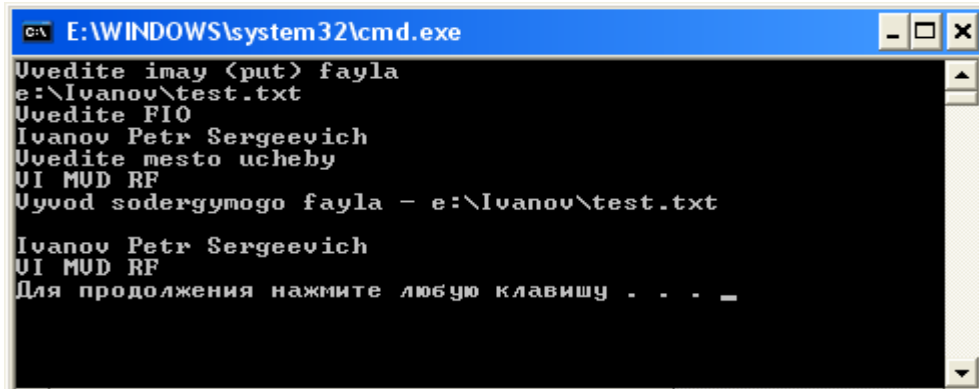
Рассмотрим примеры выполнения заданий 1, 2.

Листинг П. 1.1

```
// Laboratornay rabota N 14
/*Vypolnil stydent Ivanov P.S. */
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
    char ch, f_Name[50], str[255];
    cout << "Vvedite imay (put) fayla \n";
    cin >> f_Name;
    ofstream fout (f_Name);
    cout << "Vvedite FIO " << endl;
    cin.ignore(1, '\n'); // ubrat' symbol novoy stroki posle
                        //imeni fayla

    cin.getline(str, 255);
    fout << str << endl;
    cout << "Vvedite mesto ucheby \n";
    cin.getline(str, 255);
    fout << str << endl;
    fout.close();
    ifstream fin(f_Name);
    cout << "Vyvod sodergymogo fayla - " << f_Name << "\n";
    cout << endl;
    while (fin.get(ch)) cout << ch;
    fin.close();
    return (0);
}
```

Результат работы программы П. 1.1 представлен на рис. 1.



```
C:\ E:\WINDOWS\system32\cmd.exe
Vvedite imya (put) fayla
e:\Ivanov\test.txt
Vvedite FIO
Ivanov Petr Sergeevich
Vvedite mesto ucheby
UI MUD RF
Vывод soderzhimogo fayla - e:\Ivanov\test.txt
Ivanov Petr Sergeevich
UI MUD RF
Для продолжения нажмите любую клавишу . . . _
```

Рис. 1. Результат работы программы П. 1.1

Листинг П. 1.2.

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

int main()
{
    char ch, f_Name[50], str[255];
    cout << "Vvedite imya (put) fayla dlay otkrytiay \n";
    cin >> f_Name;
    ofstream fout (f_Name, ios :: app);
    if (fout.fail())
    {
        cout << "File " << f_Name << "ne moget byt' otkryt dlay zapisi! \n";
        return (1);
    }
    cout << "Vvedite daty rogdeniay: ";
    cin.ignore(1, '\n');
    cin.getline(str, 255);
    fout << str << endl;
    cout << "Vvedite adress: ";
    cin.getline(str, 255);
    fout << str << endl;
    fout.close();
    ifstream fin;
    fin.open(f_Name);
    if (!fin)
    {
        cout << "File " << f_Name << "ne moget byt' otkryt dlay chteniay! \n";
```

```

        return (1);
    }
    cout << endl;
    cout << "Vyvod sodergymogo fayla - " << f_Name << "\n";
    cout << endl;
    while (fin.get(ch)) cout << ch;
    fin.close();
    return (0);
}

```

Результат работы программы П. 1.2 представлен на рис. 2.

```

E:\WINDOWS\system32\cmd.exe
Uvedite imya (put) fayla dlay otlrytiay
e:\Ivanov\test.txt
Uvedite daty rogdniay: 29.11.1986
Uvedite adress: Voronezh, yl. U. Moravskay, 11

Vyvod sodergymogo fayla - e:\Ivanov\test.txt

Ivanov Petr Sergeevich
UI MVD RF
29.11.1986
Voronezh, yl. U. Moravskay, 11
Для продолжения нажмите любую клавишу . . .

```

Рис. 2. Результат работы программы П. 1.2

Пример 3. В текстовом файле mass_doub.txt хранятся вещественные числа, вывести их на экран и вычислить их количество. Результат работы программы представлен на рис. 3.

Листинг П. 1.3.

```

#include "stdafx.h"
#include <iostream>
#include <fstream>
using namespace std;
int main()
{ int n=0; double s, mass[10] ;
ifstream fin;
fin.open("mass_doub.txt");
if (!fin)
{ cout << "ошибка!\n";
  return(1); }
while (!fin.eof())
{ fin>>s;
  cout<<s<<" ";
  n++; }
fin.clear();

```

```

cout<<"\nN="<<n;
fin.seekg(0,ios::beg);
for (int i = 0; i < n; i++)
fin >> mass[i];
fin.close();
cout<<endl;
for (int i = 0; i < n; i++)
{
    cout<<mass[i]<<" ";
}
return 0; }

```

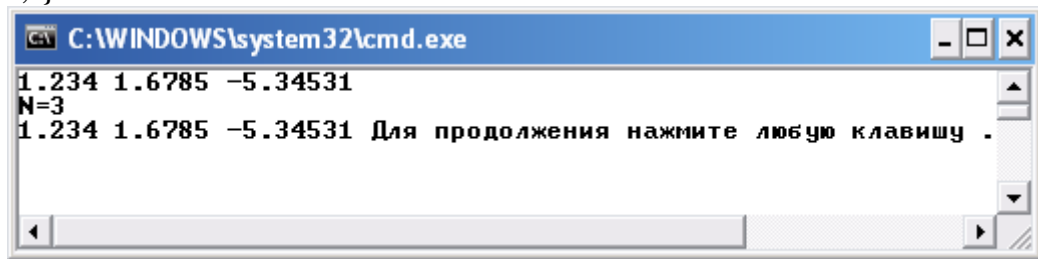


Рис. 3. Результат работы программы П. 3

Контрольные вопросы

1. Какие классы потокового ввода/вывода используют для работы с файлами C++ и в каком заголовочном файле они определены?
2. Назовите методы открытия и закрытия файлов в C++.
3. Функции состояния.
4. Как открыть файл для ввода информации? Приведите пример.
5. Как открыть файл для вывода информации? Приведите пример.
6. Как изменить стандартное поведение объекта ofstream?

Задание на самостоятельную работу

1. Подготовиться к занятию «Обработка двоичных файлов в C++».
2. Разобрать работу следующих программ:

a) #include <iostream>

#include <fstream>

int main ()

{

using namespace std;

ifstream file;

char c, c1;

file.open("basic_istream_seekg.txt");

file.seekg(2); // seek to position 2

file >> c;

cout << c << endl;

}

Содержимое файла basic_istream_seekg.txt: 0123456789

Вывод 2

б) #include <iostream>

#include <fstream>

int main()

{

using namespace std;

ifstream file;

char c;

streamoff i;

file.open("basic_istream_tellg.txt");

i = file.tellg();

file >> c;

cout << c << " " << i << endl;

i = file.tellg();

file >> c;

cout << c << " " << i << endl;

}

Содержимое файла basic_istream_tellg.txt: 0123456789

Вывод

0 0

1 1

```
в) #include <iostream>
using namespace std;
```

```
int main()
{
    char c[10];
    int count = 5;

    cout << "Type 'abcde': ";

    // Note: cin::read is potentially unsafe, consider
    // using cin::_Read_s instead.
    cin.read(&c[0], count);
    c[count] = 0;

    cout << c << endl;
}
```

Вывод на экран: Type «abcde»: abcde

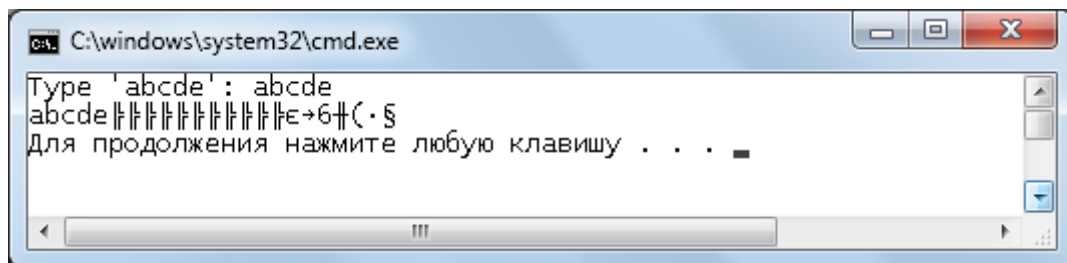
Ввод abcde

Вывод abcde

Примечание, если закомментировать строку

```
// c[count] = 0;
```

то результат будет другой:



```
г) #include <iostream>
using namespace std;
```

```
int main( )
{
    char c[10], c2;
    cout << "Type 'abcde': ";

    c2 = cin.peek( );
    cin.getline( &c[0], 9 );

    cout << c2 << " " << c << endl;
}
```

Вывод на экран: Type «abcde»: abcde

Ввод abcde

Вывод a abcde