



External File Name and File Closing

External file name is the one used by the OS (the real file name)

```
in_stream.open( "a:\\grades.dat" );
```

- typically used only once in the program: when it is opened
- file is always referred to by the **stream** it is connected to

```
in_stream >> grade;
```

Every file for I/O should be closed when the program is finished

- disconnect the stream from the file

```
in_stream.close();  
out_stream.close();
```

- system will close the file if program ends normally, if not closed within the program
- closing files in the program is encouraged

in case the program ends abnormally

for changing the use of files (e.g. write a file, then read it back in again)



Useful Functions for File I/O

- `fail`: to test whether a stream operation has failed

```
in_stream.open("grades.dat");  
if (in_stream.fail())  
{  
    cout << "Input file opening failed"  
        << endl;  
    exit(1);  
}
```

- `exit` is a predefined function to terminate the program immediately
- `exit` takes a single integer parameter: 1(error), 0(otherwise)
- To use `exit`, the program must include

```
#include <stdlib.h>;
```



Formatting Output with Stream Functions

- Remember the magic formula?

```
cout.setf(ios::fixed);  
cout.setf(ios::showpoint);  
cout.precision(2);
```

- They can be used on any output file stream

```
out_stream.setf(ios::fixed);  
out_stream.setf(ios::showpoint);  
out_stream.precision(2);
```

- `setf`: set flag for doing something in one of the two possible ways, depending on the kind of flag (page 231)

```
ios::fixed -- use fixed point instead of e-notation  
ios::showpoint -- include a decimal point
```



Streams as Arguments to Functions

Streams can be arguments (actual parameters) to a function the formal parameters must be a reference parameters

```
#include <iostream.h>
#include <fstream.h>
void make_square(ifstream& data_file,
                ofstream& sq_file);

int main()
{
    ifstream fin;
    ofstream fout;
    fin.open("data.dat");
    fout.open("square.dat");
    make_square(fin, fout);
    fin.close(); fout.close();
    cout << "Program finished." << endl;
    return 0 ;
}

void make_square(ifstream& data_file,
                ofstream& sq_file)
{
    double next;
    while (data_file >> next)
    {
        sq_file << next * next << endl;
    }
    return ;
}
```



Checking the End of a File

- A program may need to read all the data from a file without knowing how many numbers are in the file

```
while (there are still numbers)
{
    data_file >> next;
    sq_file << next * next << endl;
}
```

- An expression involving >> is both an action (extraction) and a logical condition

```
while (data_file >> next)
{
    sq_file << next * next << endl;
}
```



Character Input/Output

- All data is input and output as character data

10 is actually two characters '1' and '0'

--- conversion is done automatically

- Member functions for character data I/O is provided:

get: read in one character and store it in a variable

put: output one character

```
char next_symbol;
```

```
cin.get(next_symbol);
```

```
cout.put('a');
```

- How is get different from extraction operator (>>)?

some things are done automatically when using >> while nothing is done automatically with get

```
char c1, c2, c3;
```

```
cin.get(c1);
```

```
cin.get(c2);
```

```
cin.get(c3);
```

```
your input: A BC
```

```
D
```



Character Input using get

- Careful for blanks and new-line characters

```
int number1; double number2 ;  
char symbol1, symbol2;  
cout << "Enter two numbers:\n";  
cin >> number1 >> number2;  
cout << "Enter two letters:\n";  
cin.get(symbol1);  
cin.get(symbol2);
```

```
program execution: Enter two numbers:  
                   22 3.5  
  
                   Enter two letters:  
                   2A
```

- get does not skip over line breaks and spaces



Member Function eof

- Used to determine when all of the file has been read
- Works best when input file consists of text

```
ifstream fin;  
fin.open("input.txt");  
char next;  
fin.get(next);  
while (! fin.eof())  
{  
    cout << next;  
    fin.get(next);  
}
```



Predefined Character Functions

For text processing, several predefined functions are provided

- to convert lowercase letter to uppercase or vice versa

```
char symbol = toupper('a');  
cout << symbol;
```

- to check whether the character is space, alphabet, digit, etc

```
char symbol;  
cin.get(symbol);  
if (isspace(symbol))  
{  
    cout << "It is a whitespace."  
}
```

- to use them, a program must contain

```
#include <ctype.h>
```

some of the predefined functions are displayed in p. 259



File -- Character Output

```
#include <iostream.h>
#include <fstream.h>
int main ()
{
    int i = 21;
    char chX = 'X', chY = 'Y';
    char chZ = 'Z', che = '\n';

    ofstream fout;
    fout.open("a:\\my.dat");

    fout << "i= " << i << endl;
    fout.put(chX);
    fout.put(chY);
    fout.put(che);
    fout.put(chZ);
    fout.close();
    return 0;
}
```

```
i= 21
XY
Z
```



File Input

```
#include <iostream.h>
#include <fstream.h>
int main ()
{
    int i = 0;
    char ch = 'A';

    ifstream fin;
    fin.open("a:\\my.dat");

    fin.get(ch);
    while ( !fin.eof() )
    {
        cout << ch;
        i++;
        fin.get(ch);
    }
    cout << endl << i << endl;
    fin.close();
    return 0;
}
```

```
    i= 21
    XY
    Z
    10
```




Simple (One-Dimensional) Arrays

- Contiguous block of storage (elements)
- Represented by *one* name
- Individual elements accessed through *subscripts*
- All elements have same type
- example:

```
int score[3];  
score[0] = 10;  
score[1] = 15;  
score[2] = 20;
```

```
cout << score[1]; //prints 15
```

- Subscripts start with 0 (zero) in arrays

The index type is integer and the index range must be 0 ... n-1

- Element can be *any* type — the *base type*
- Analogs: egg carton or apartments



Example

```
#include <iostream.h>

int main()
{
    int score[7];
    int i = 0;
    while (i < 7)
    {
        cin >> score[i];
        cout << score[i] << endl;
        i++;
    }
    return 0;
}
```

First number score[0]
Second number score[1]
.....
.....
Seventh number score[6]