Chapter 13

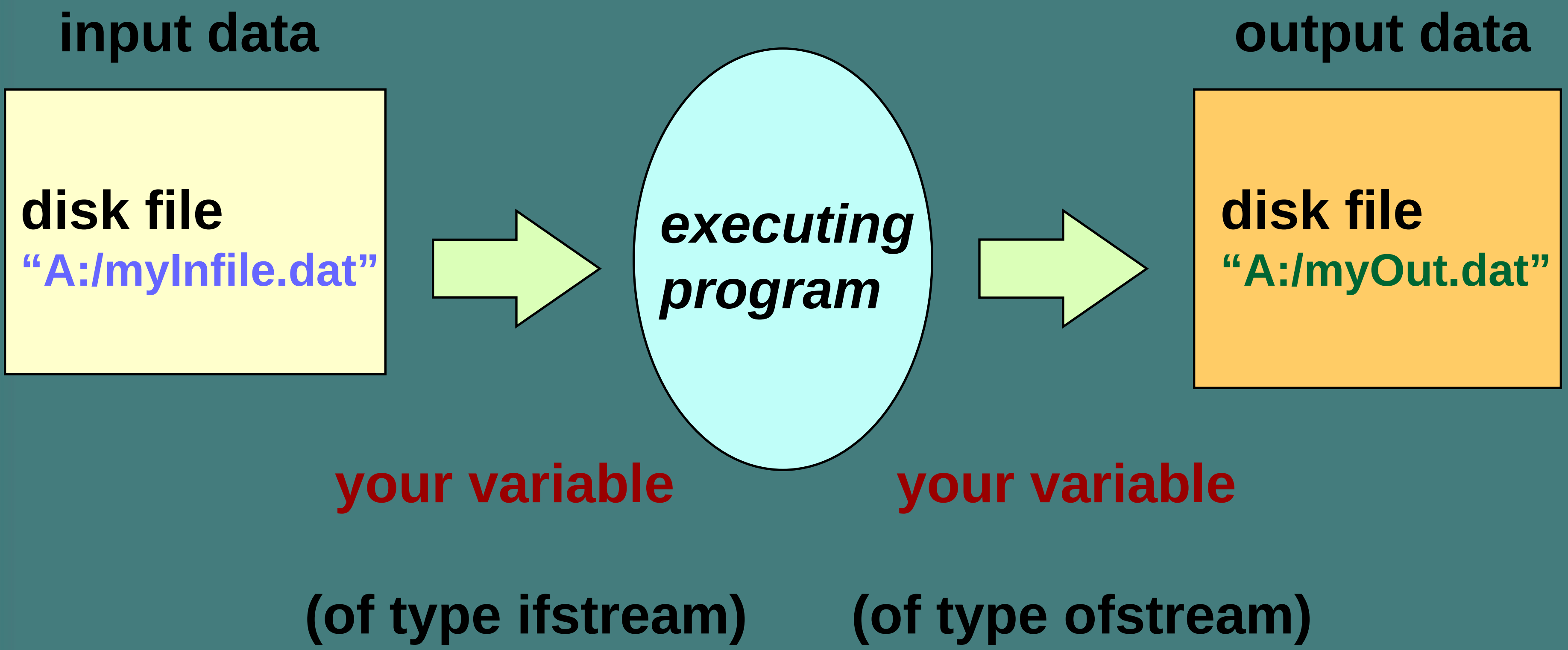
Input/Output Stream

Chapter 13 Topics(part 2)

- Using Data Files for Input and Output
  - Diskette Files for I/O
  - Input and Output ASCII Files
  - Input and Output Binary Files
  - A Case Study

Diskette Files for I/O

#include <fstream>



input data

disk file
"A:/myInfile.dat"

executing program

output data

disk file
"A:/myOut.dat"

your variable

(of type ifstream)

your variable

(of type ofstream)

To Use Disk I/O, you must

- use #include <fstream>
- choose valid identifiers for your filestreams and declare them
- open the files and associate filestream identifiers with disk names
- use your filestream identifiers in your I/O statements
- close the files

Statements for Using Disk I/O

```
#include <fstream>

ifstream   myInfile;                          // declarations
ofstream  myOutfile;

myInfile.open(“A:/myIn.dat”);                 // open files
myOutfile.open(“A:/myOut.dat”);

myInfile.close( );                            // close files
myOutfile.close( );
```

What does opening a file do?

- associates the C++ identifier for your file with the physical (disk) name for the file
- if the input file does not exist on disk, open is not successful
- if the output file does not exist on disk, a new file with that name is created
- if the output file already exists, it is erased
- places a *file reading marker* at the very beginning of the file, pointing to the first character in it

Stream Fail State

- when a stream enters the fail state, further I/O operations using that stream have no effect at all. But the computer does not automatically halt the program or give any error message

- possible reasons for entering fail state include:
  - invalid input data (often the wrong type)
  - opening an input file that doesn’t exist
  - opening an output file on a diskette that is already full or is write-protected

Chapter 13 Topics(part 2)

- Using Data Files for Input and Output
  - Diskette Files for I/O
  - Input and Output ASCII Files
  - Input and Output Binary Files
  - A Case Study

Output ASCII Files

```
#include<iostream>
#include <fstream>
using namespace std;
int main( )
{  int a[10];
   ofstream outfile("f1.dat"); //
   if(!outfile)                          //
    { cerr<<"open error!"<<endl;
       exit(1);}
   cout<<"enter 10 integer numbers:"<<endl;
   for(int i=0;i<10;i++)
    {  cin>>a[i];
       outfile<<a[i]<<" ";}           //
   outfile.close();                   //
   return 0;
}
```

Input ASCII Files

```
#include<iostream>
#include <fstream>
using namespace std;
int main( )
{  int a[10];
   ifstream infile("f1.dat");   //
   if(!infile)                        //
  {  cerr<<"open error!"<<endl;
     exit(1);}
  cout<<"10 integer numbers are:"<<endl;
  for(int i=0;i<10;i++)
  {  infile>>a[i];      //
     cout<<a[i]<<" ";}
  infile.close();                   //
  return 0;
}
```

Chapter 13 Topics(part 2)

- Using Data Files for Input and Output
  - Diskette Files for I/O
  - Input and Output ASCII Files
  - Input and Output Binary Files
  - A Case Study

Using write( )for Output

```
#include<iostream>
#include <fstream>
using namespace std;
struct student
{char name[20];
 int num;
 int age;
 char sex;
};
int main( )
{  student stud[3]={"Li",1001,18,'f',"Feng",1002,19,'m',"Wang",1004,17,'f'};
   ofstream outfile("stud.dat",ios::binary);
   if(!outfile)
      {cerr<<"open error!"<<endl;
       exit(1); }
   for(int i=0;i<3;i++)
   outfile.write((char *)&stud[i],sizeof(stud[i]));
   outfile.close( );
   return 0;
}
```

Using read( ) for Input

```
#include<iostream>
#include <fstream>
using namespace std;
struct student
{char name[20];
 int num;
 int age;
 char sex;
};
```

```
int main( )
{  student stud[3];
   int i;
   ifstream infile("stud.dat",ios::binary);
   if(!infile)
    {  cerr<<"open error!"<<endl;
       abort( );}
   for(i=0;i<3;i++)
   infile.read((char *)&stud[i],sizeof(stud[i]));
   infile.close( );
   for(i=0;i<3;i++)
   {  cout<<"NO."<<i+1<<endl;
      cout<<"name:"<<stud[i].name<<endl;
      cout<<"num:"<<stud[i].num<<endl;
      cout<<"age:"<<stud[i].age<<endl;
      cout<<"sex:"<<stud[i].sex<<endl<<endl;}
   return 0;
 }
```

Random Access to Binary Files

```
#include<iostream>
#include <fstream>
using namespace std;
struct student
{int num;
 char name[20];
 float score;
};
```

```
int main( )
{  student stud[5]={1001,"Li",85,1002,"Feng",97.5,
        1004, "Wang",54,1006,"Tan",76.5,1010,"lin",96};
   fstream iofile("stud.dat",ios::in|ios::out|ios::binary);
   if(!iofile)
    {cerr<<"open error!"<<endl;
     abort( );}
   for(int i=0;i<5;i++)
     iofile.write((char *)&stud[i],sizeof(stud[i])); //
   student stud1[5];                    //
   for(i=0;i<5;i=i+2)
   {  iofile.seekg(i*sizeof(stud[i]),ios::beg);  //
      iofile.read((char *)&stud1[i/2],sizeof(stud1[0])); //
      cout<<stud1[i/2].num<<“ ”<<stud1[i/2].name<<“   ”
              <<stud1[i/2].score<<endl;}
   cout<<endl;
```

Random Access to Binary Files(Cont.)

```
stud[2].num=1012;                          //
 strcpy(stud[2].name,"Wu");
 stud[2].score=60;
 iofile.seekp(2*sizeof(stud[0]),ios::beg);   //
 iofile.write((char *)&stud[2],sizeof(stud[2])); //
 iofile.seekg(0,ios::beg);                      //
 for(i=0;i<5;i++)
   {iofile.read((char *)&stud[i],sizeof(stud[i]));  //
    cout<<stud[i].num<<" "<<stud[i].name<<" "<<stud[i].score<<endl;
   }
 iofile.close( );
return 0;
}
```

Chapter 13 Topics(part 2)

- Using Data Files for Input and Output
  - Diskette Files for I/O
  - Input and Output ASCII Files
  - Input and Output Binary Files
  - A Case Study

Map Measurement Case Study

You want a program to determine walking distances between 4 sights in the city. Your city map legend says one inch on the map equals 1/4 mile in the city. Read from a file the 4 measured distances between sights on the map and the map scale.

Output to a file the rounded (to the nearest tenth) walking distances between the 4 sights.

Using File I/O

```
// ***************************************************
//  Walk program using file I/O
//  This program computes the mileage (rounded to nearest
//  tenth of mile) for each of 4 distances, using input
//  map measurements and map scale.
// ***************************************************

#include <iostream>         // for cout, endl
#include <iomanip>          // for setprecision
#include <iostream>         // for file I/O

using namespace std;

float  RoundToNearestTenth( float );  // declare function
```

```
int  main( )
{
     float     distance1;        // First map distance
     float     distance2;        // Second map distance
     float     distance3;        // Third map distance
     float     distance4;        // Fourth map distance
     float     scale;            // Map scale (miles/inch)

     float     totMiles;         // Total of rounded miles
     float     miles;            // One rounded mileage

     ifstream  inFile;           // First map distance
     ofstream  outFile;          // Second map distance

     outFile << fixed << showpoint  // output file format
             << setprecision(1);
                                 // Open the files
     inFile.open(“walk.dat”);

     outFile.open(“results.dat”);
```

```
                            // Get data from file

inFile >>  distance1  >> distance2  >> distance3
       >>  distance4  >> scale;

totMiles = 0.0;             // Initialize total miles

        // Compute miles for each distance on map


miles = RoundToNearestTenth( distance1 * scale );

outFile <<  distance1 << “ inches on map is “
        <<  miles  << “ miles in city.” << endl;

totMiles = totMiles + miles;
```

```
miles = RoundToNearestTenth( distance2 * scale );

outFile <<  distance2 << “ inches on map is “
        <<  miles  << “ miles in city.” << endl;

totMiles = totMiles + miles;
miles = RoundToNearestTenth( distance3 * scale );

outFile <<  distance3 << “ inches on map is “
        <<  miles  << “ miles in city.” << endl;

totMiles = totMiles + miles;


miles = RoundToNearestTenth( distance4 * scale );

outFile <<  distance4 << “ inches on map is “
        <<  miles  << “ miles in city.” << endl;

totMiles = totMiles + miles;
```

```
     //  Write total miles to output file

     outFile <<  endl << “Total walking mileage is  “
             <<  totMiles  <<  “ miles.”  << endl;

     return 0 ;                  //  Successful completion
}

// ***************************************************

float  RoundToNearestTenth ( /* in */  float floatValue)

//  Function returns floatValue rounded to nearest tenth.

{
     return  float(int(floatValue * 10.0 + 0.5)) / 10.0;
}
```