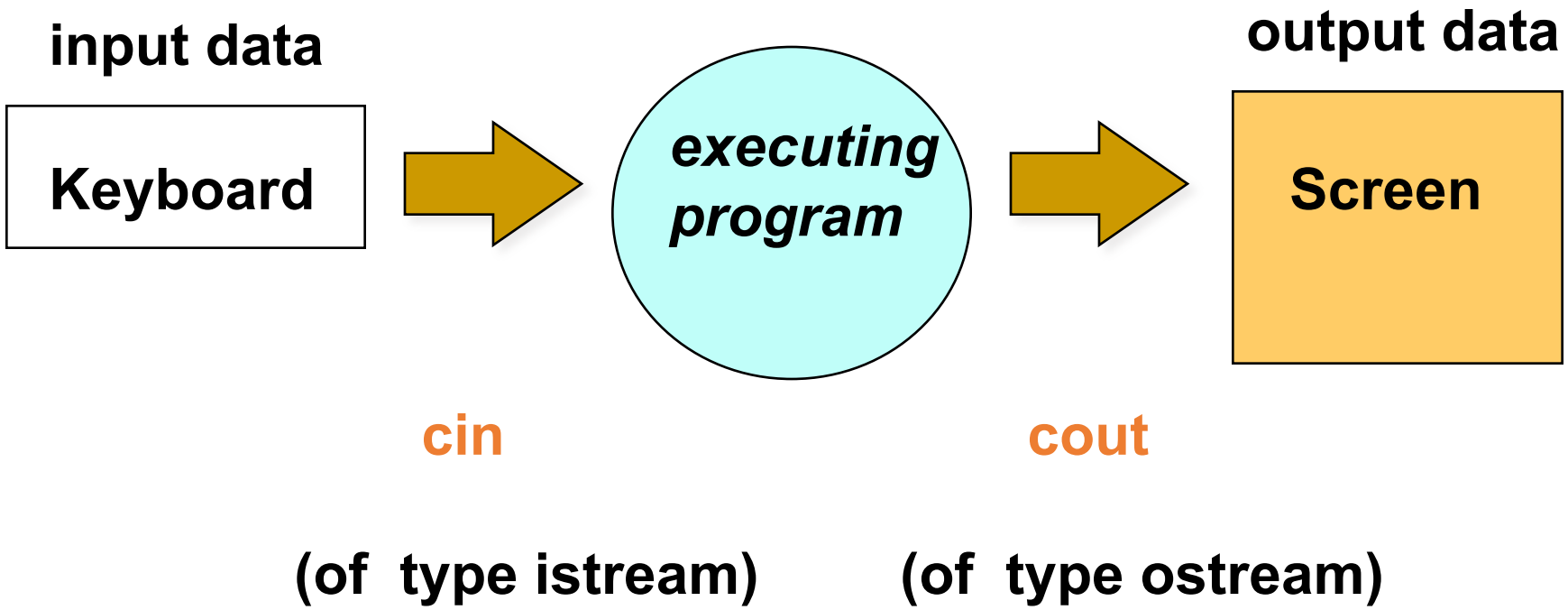
C++ Review

Keyboard and Screen I/O

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



namespace

- In slides that follow, assume the statement:

  using namespace std;

- We explain namespace in Chapter 2

<iostream> is header file

- A library that defines 3 objects:
  - An istream object named cin (keyboard)
  - An ostream object named cout (screen)
  - An ostream object named cerr (screen)

Insertion Operator (<<)

- **The insertion operator << takes 2 operands.**
- **The left operand is a stream expression, such as cout.**
- **The right operand is an expression describing what to insert into the output stream. It may be of simple type, or a string, or a manipulator (like endl).**

Extraction Operator (>>)

- **Variable cin is predefined to denote an input stream from the standard input device (the keyboard).**
- **The extraction operator >> called "get from" takes 2 operands. The left operand is a stream expression, such as cin. The right operand is a variable of simple type.**
- **Operator >> attempts to extract the next item from the input stream and store its value in the right operand variable.**

Extraction Operator >>

"skips" (reads but does not store anywhere)

leading whitespace characters

(blank, tab, line feed, form feed, carriage return)

before extracting the input value from the stream (keyboard or file).

To avoid skipping, use function get to read the next character in the input stream.

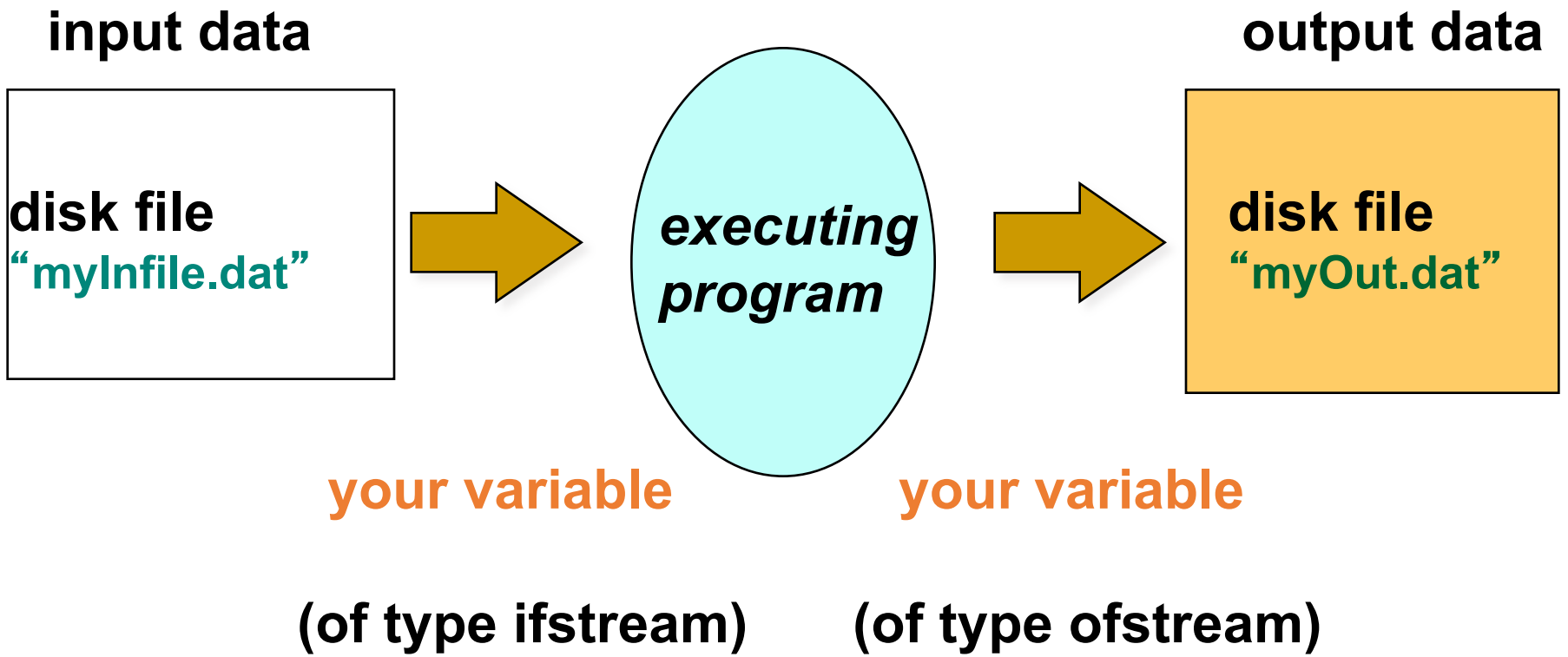
```
cin.get(inputChar);
```

```
#include  <iostream>
int  main( )
{                                  // USES KEYBOARD AND SCREEN I/O
  using namespace std;
  int       partNumber;
  float      unitPrice;

  cout  <<  "Enter part number followed by return : "
        <<  endl ; // prompt
  cin   >>  partNumber ;
  cout  <<  "Enter unit price followed by return : "
        <<  endl ;
  cin   >>  unitPrice ;
  cout  <<  "Part # "  <<  partNumber  // echo
        <<  "at  Unit Cost: $ "  <<  unitPrice
        <<  endl ;
     return 0;
}
```

Disk files for I/O

```
#include <fstream>
```



For File I/O

- use #include <fstream>
- choose valid variable identifiers for your files and declare them
- open the files and associate them with disk names
- use your variable identifiers with >> and <<
- close the files

Statements for using file I/O

```
#include <fstream>
using namespace std;
ifstream   myInfile;      // declarations
ofstream  myOutfile;

myInfile.open("myIn.dat"); // open files
myOutfile.open("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

```
#include  <fstream>
int  main( )
{                                   // USES FILE I/O
  using namespace std;
  int  partNumber;
  float     unitPrice;
  ifstream   inFile;                // declare file variables
  ofstream   outFile;

  inFile.open("input.dat");       //open files
  outFile.open("output.dat");

  inFile  >>  partNumber ;
  inFile  >>  unitPrice ;
  outFile <<  "Part # "  <<  partNumber  // echo
          <<  "at  Unit Cost: $ "  << unitPrice << endl ;
   return   0;
}
```

Stream Failure

- **When a stream enters the fail state, further I/O operations using that stream are ignored. 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 disk that is already full or is write-protected.**

```
#include  <fstream>
#include  <iostream>
using namespace std;
int  main( )
{                    // CHECKS FOR STREAM FAIL STATE

  ifstream   inFile;
  inFile.open("input.dat");      // try to open file
  if ( !inFile )
  {
    cout  << "File input.dat could not be opened.";
     return  1;
  }
            . . .
  return   0;
}
```