# Streams

# Input and output streams

- C++ input and output is based upon the concept of streams
- Streams are a family of classes that provide generic methods for a variety of functionality e.g. screen, keyboard, files and memory
- 3 streams are open by default
  - cout: by default the screen
  - cerr: by default the screen
  - cin: by default the keyboard
- Input (>>) and output (<<) operators are defined for default types/objects and can be overloaded to work with user defined objects

# cin, cout and cerr

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

void main()
{
   int i;
   char buff[100];
   cout << “Enter a name and a number, but not -1” << endl;
   cin >> ws >> buff >> ws >> i;
   if (i==-1)
        cerr << “I told you not to!!!!/n” << flush;
}
```

# files

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

void main()
{
    ifstream in(“in.txt”);
    ofstream out(“out.txt”);

    if (in==NULL)
          cerr << “can open file” << endl;
    if (!out)
          cerr << “can open file” << endl;


    for(int i(0);i<10;i++){
      in >> num;
      if (in.eof())
        cerr << “reached the end of file before I finished” << endl;
        out << num << endl;
    }
    in.close();
    out.close();
}
```

# files

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

void main()
{
   // Files for both input and output or appending as
   opened as fstream

   fstream myfile;
   myfile.open(“data.txt”, ios::in);  // input
   //                      ios::out); // output
   //                      ios::app); // append
```

# getline and strings

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

int main ()
{
  string mystr;
  cout << "What's your name? ";

  getline (cin, mystr);

  cout << "Hello " << mystr << ".\n";
  cout << “Where do you live? ";

  cin.getline(mystr,100);

  cout << "I live in" << mystr << " too!\n";
  return 0;
}
```

# getline and strings

```
// stringstreams
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

int main ()
{
  string mystr;
  float price=0;
  int quantity=0;

  cout << "Enter price: ";
  getline (cin,mystr);
  stringstream(mystr) >> price;
  cout << "Enter quantity: ";
  getline (cin,mystr);
  stringstream(mystr) >> quantity;
  cout << "Total price: " << price*quantity << endl;
  return 0;
}
```

```
string mystr ("1204");
int myint;
stringstream(mystr) >> myint;
```

# Other useful methods

```
in.get(c);      // Reads a character c from istream in
out.put(c);     // Writes a character c to ostream out
in.putback(c); // Puts a character c back on the front of istream in
in.peek();      // Returns (as an int) the next character of istream
                // in without removing it
in.eof();       // Returns true if end of file is reached

cout.width(5);                                  // Set field width to 5
cout.setf(ios::left, ios::adjustfield);         // Left justify
cout << '(' << -7 << ")\n";                     // (-7   )


cout << setw(13) << " Hello " << setw(4) << 4 << endl;
```

# Other useful methods

```
cout.setf(ios::left,ios::adjustfield);     // left
cout.setf(ios::right,ios::adjustfield);    // right

ios::skipws     // skips whitspace on input
ios::left       // left justification
ios::right      // right justifiction
ios::internal   // pads after sign or base character
ios::dec        // decimal format for integers
ios::oct        // octal format for integers
ios::hex        // hex format for integers
ios::showbase   // show the base character for octal or hex
ios::showpoint  // show the decimal point for all floats
ios::uppercase  // uppercase A-F for hex
ios::showpos    // show +ve sign for numbers
ios::scientific // use exponential notation
ios::fixed      // used oridnary decimal notation
ios::unitbuf    // flush the buffer

ios::basefield    = ios::dec | ios::oct | ios::hex
ios::adjustfield  = ios::left | ios::right | ios::internal
ios::floatfield   = ios::scientific | ios::fixed
```

# Other useful methods

```
// Field width
cout.width(20);
// or, using a manipulator:
cout << setw(20);
// These only change the width for the next item output only.

// Fill character
cout.fill('0');
// or
cout << setfill('0');

// Justification
cout.setf(ios::right, ios::adjustfield);

// Precision
// The format of floats can be set by:
cout.setf(ios::scientific, ios::floatfield);
// and the number of decimal places by:
cout.precision(2);
// or:
cout << setprecision(2);
```