Object oriented programming

- What is Class? What is Object?
  - From object-oriented point of view
    - ***Class*** is a user-defined data type which contains relevant data and functions
    - ***Object*** is a variable declared under some class data type
  - From philosophy concept
    - Class is an abstract concept that describes the attributes of a collection of objects

From C to C++

- Namespace
  - 變數、函數、或物件所屬的空間名稱,在不同的namespace中可使用相同的變數名稱。
  - std: C++全部系統提供的函數所屬的namespace
  - avoid conflict of variable names in the different class libraries

namespace example

```
//This program outputs the message
//
//C++:one small step for the program,
//one giant leap for the programmer
//
//to the screen
#include <iostream>
using namespace std;
int main(){
cout <<"C++:one small step for the program,\n"
        <<"one giant leap for the programmer \
return 0;
}
```

compare to C:

- #include <stdio.h>
- main( ){
- printf(“….”);
- without namespace

namespaces

- create namespace
  - examples:

```
namespace  mfc{
   int  inflag;
   void  g(int);
   …
 }

namespace  owl{
   int inflag;
   …
}
```

namespace

- use variables in a namespace
  - use scope resolution operator **::**
  - e.g.

```
mfc::inflag = 3;
owl::inflg = -823;
cout << mfc::inflag;
```

- the **using** directive
  - e.g..

```
using  namespace mfc;
inflag = 3;              // 相當於  mfc::inflag = 3;
owl::inflag = -823;
```

C++ Input/Output

- C++ Input/Output objects
  - cin standard input
  - cout standard output
  - cerr standard error

  - e.g.

```
cin >> x;
cin >> len;
cout << x;
cout << len;
```

```
cin >> x >> len;

cout << x <<  len;
```

C++ Input/Output

- example

```
#include <iostream>
using namespace std;
int main( ) {
int  id;
float   av;
cout << “Enter the id and the average:”;
cin >> id >> av;
cout << “Id:” << id << ‘\n’
        << “Average:” << av << ‘\n’;
return 0;
}
```

```
Enter the id and the average:900038 90.8
Id:900038
Average:90.8
```

C++ Input Output

- Manipulators
  - for the format of I/O
  - set width to n: setw(n)

```
for (i=1; i<=1000; i*=10)
  cout << setw(6) << i << '\n';
```

```
     1
    10
   100
  1000
```

manipulators

- endl: end of line, write new line
  - e.g.

```
int  i=4, j=6, k=8;
char  c='!';
cout << i << c << endl
        << j << c <<'\n'
        << k << c << endl;
```

```
4!
6!
8!
```

manipulators

- oct (octal), hex(hexadecimal), dec(decimal)
  - e.g.

```
int  i = 91;
cout << “i=“ << i << “ (decimal)\n”;
cout << “i=“ << oct << i << “ (octal)\n”;
cout << “i=“ << hex << i << “ (hexadecimal)\n”;
cout << “i=“ << dec << i << “ (decimal)\n”;
```

```
i=91 (decimal)
i=133 (octal)
i=5b (hexadecimal)
i=91 (decimal)
```

manipulators

- listed in chap. 14-9
  - dec, endl, fixed, flush, hex, left, oct, right, scientific, setfill(c), setprecision(n), setw(n), showpoint, noshowpoint, showpos, noshowpos, skipws, noskipws, ws
  -

manipulators

- setfill, setprecision
  - e.g.

```
float  a = 1.05, b=10.15, c=200.87;
cout << setfill('*') << setprecision(2);
cout << setw(10) << a << '\n';
cout << setw(10) << b << '\n';
cout << setw(10) << c << '\n';
```

```
******1.05
*****10.15
****200.87
```

files (example)

```
#include <fstream>
using namespace std;
const int cutoff =6000;
const float rate1 =0.3;
const float rate2 =0.6;
int main(){
ifstream infile;
ofstream outfile;
int income,tax;
infile.open("income.txt
  ");
outfile.open("tax.txt")
  ;
while (infile >>income
  ){
    if (income <cutoff
  )
tax =rate1 *income;
else
tax =rate2 *income;
outfile<<"Income="<<income
          <<"greenbacks \n"
          <<"Tax ="<<tax
          <<"greenbacks \n";
}
infile.close();
outfile.close();
return 0;
}
```

files (example cont.)

input file “income.txt”

2214 10500 31010

result:

output file “tax.txt”

Income = 2214 greenbacks
Tax= 664 greenbacks
Income = 10500 greenbacks
Tax= 6299 greenba cks
Income = 31010 greenbacks
Income = 18605 greenbacks

files

- testing whether files are open
  - file object converts to ‘true’ if open successfully, otherwise converts to ‘false
  - e.g.

```
ifstream  infile;
ifstream.open(“scores.dat”);
…
if (infile) {   …    }     //  if open sucessfully
or
if (!infile) { …     }     //  if fail to open the file
```

files

- e.g..

```
ifstream  infile;
infile.open(“scores.dat”);
if (!infile) {
   cerr << “Unable to open scores.dat\n”;
   exit(0);
}
```

C++ features

- bool data type
  - values: true (1) or false(0)
  - manipulators: boolalpha, noboolalpha (default)
  - e.g.

```
bool  flag;
flag = (3 < 5);
cout << flag << '\n';
cout << boolalpha << flag << '\n';
```

```
true
```

the type string

- string initialization
  - e.g.

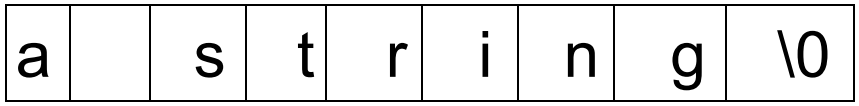
```
#include <string>
string s1;
string s2 = “Bravo”;
string s3 = s2;
string s4(10,’x’);
cout << s1 << ‘\n’;
cout << s2 << ‘\n’;
cout << s4 << ‘\n’;
```

```
Bravo
xxxxxxxxxx
```

the type string

- C-style string (end with a null char '\0')

```
char* mystring = "a string";
    or
char  mystring[ ] = "a string";
printf("%s\n", mystring);
```



char mystring[9]

- the null character '\0' is added by the C compiler automatically

the type string

- string length

```
string  s = “Ed  Wood”;
cout  << “Length=“ << s.length( ) << ‘\n’;
```

```
Length=7
```

- input a string
  - separate by space or new line

```
cout << “Enter a string:”;
cin >> s;
cout << s;
```

```
Ed  Wood
Ed
```

getline function example (copy file)

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main(){
string buff;
ifstream infile;
ofstream outfile;
cout << “Input file name:”;
cin >> buff;
infile.open(buff.c_str());
cout << “Output file name:”;
cin >> buff;
outfile.open(buff.c_str());
while (getline(inflie, buff))
    outfile <<buff<<“\n\n”;
infile.close();
outfile.close();
return 0;
}
```