函式及運算子的多面性 (Overloading)

函式的複載 (Overloading)

- 使一個算符或函式具有處理多種資料型別能力的方法稱為複載(Overloading)
- C++中對複載的意義為:讓同一種名稱或用法具有多種意義
- 當我們定義多個具有相同名稱,但卻
  - 有不同參數個數或
  - 相同參數個數但參數型別不同

  時, 這就稱為函式的複載

```
int    max(int  a, int  b){ ….}
char max(char a, char b) {….}
long max(long a,long b) {….}

int  i=max(2,4);
char c=max('a','b');
long l=max(23L,56L);
```


- 函式傳回值的型別以及函式的參數名稱不可作為複載函式的識別之用

```
int    print();
long print();  // error

int print(int  a);
int print(int  b);  // error
```

- 由於 typedef 並不會定義出新的型別 (只是製造出一別名而已),因此,用 typedef 所定義出的型別仍以其原始型別為複載之依據

```
Typedef  char flag;

print(char);
print(flag);  // error
```

- 不同的 scope 各有其獨自的 overloading 空間

```
Func1()                          Func2()
{  int max(int a, int b);        {  int max(char a, char b);
   …. }                             …. }
```

```
// Overloading 使用範例

#include <iostream.h>
void  repchar();
void  repchar(char0;
void  repchar(char,  int);

void main()
{
  repchar();
  repchar('=');
  repchar('+',30);
}

void repchar()
{
  for ( int j=0;j<45;j++)
       cout << '*';
  cout << endl;
}
```

```
void repchar(char ch)
{
   for (int j=0;j<45;j++)
        cout << ch;
   cout << endl;
}

void repchar(char ch, int n)
{
   for (int j=0; j<n; j++)
       cout << ch;
   cout << endl;
}

Output:
****…***
===…=====
++… ++
```



```
// Overloading 使用範例

#include <iostream.h>
#include <conio.h>
#include <process.h>  //for
exit(1)
const int Max=100;
class Stack
{ protected:
   int st[Max];
   int top;
  public:
    Stack(){top=0;}
    void Push(int var)
      { st[++top]=var; }
    int Pop() { return st[top--]; }
};
```

```
class Stack2: public Stack
{ public:
   void Push(int var)
    { if (top<Max)
        Stack::Push(var);
      else
        { cout << "\nError: stack is
                        full";
          exit(1); }
    }
   int Pop()
    { if (top>0)
        return Stack::Pop();
      else
        { cout << "\nError: stack is
                        empty";
          exit(1); }
    }
};
```

```
void main()
{ clrscr();
  Stack2 s2;
  s2.Push(11);
  s2.Push(22);
  s2.Push(33);
  cout << endl << s2.Pop();
  cout << endl << s2.Pop();
  cout << endl << s2.Pop();
  cout << endl << s2.Pop();
}
```

```
Output:
33
22
11
Error: stack is empty
```

運算子的多面性 (Operator Overloading)

☐ C++與C相同,提供算術運算子(+,-,*,/,++,--…)及關係運算子(>,>=..)以及算術指定運算子(+=,*=,…) 能對基本資料型態如 int, float, long 等執行運算

如: int a=b+c;
league+=le;

☐對使用者自行定義較複雜的資料型態如結構或類別,就不能拿這些運算子直接作運算,例如,不能直接將兩個屬於 room 的類別變數作相加來當作其成員變數相加,必須另外透過成員函數進行運算.

```
...
Class room
{  private:
      float  ledge, sedge;
   public:
      …
      void  addsquare(room r1, room r2)
}  r1, r2, r3;
...
```

```
void  room :: addsquare(room  r1, room r2)
{
   ledge=r1.ledge + r2.ledge;
   sedge=r1.sedge+ r2.sedge;
}
…
void  main()
{ ...
  r3.addsuare(r1, r2);
  r3=r1+r2;  //error }
```

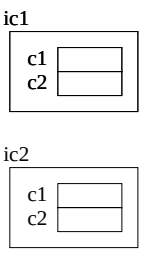

☐ C++提供運算子的複載(多元定義 operator overloading), 允許使用者對運算子重新定義, 經重新定義的運算子也能對複雜的資料型態進行類似的運算如 r3=r1+r2;

☐ 對使用者自行定義較複雜的資料型態如結構或類別,就不能拿這些運算子直接作運算,例如,不能直接將兩個屬於 room 的類別變數作相加來當作其成員變數相加,必須另外透過成員函數進行運算.

```
#include <iostream.h>
// page 12-5
// 單一運算元:無參數,無 return值
class incount
{  private:
     int c1,c2;
   public:
     incount()
     { c1=0; c2=1000; }

     void retcount(void)
     { cout << "c1=" << c1 << endl;
       cout << "c2=" << c2 << endl;}
```

```
void operator ++()
     { c1++;
       c2++; }
};
void main()
{ incount ic1,ic2;
  ic1.retcount();
  ic2.retcount();
  ic1++; // considered as ic1.++()
  ic2++; // considered as ic2.++()
  ic1.retcount();
  ic2.retcount();
 }
```



```
#include <iostream.h>
// page 12-8
// 單一運算元:無參數,有 return值
class incount
{  private:
     int c1,c2;
   public:
     incount()
     { c1=0; c2=1000; }
   void retcount(void)
     { cout << "c1=" << c1 << endl;
       cout << "c2=" << c2 << endl;}
   incount operator ++()
     { c1++; c2++;
       incount temp;
       temp.c1=c1; temp.c2=c2;
       return temp; }
};
void main()
{ incount ic1,ic2;
  ic1.retcount();
  ic2.retcount();
  ic1++; // considered as ic1.++
  ic1.retcount();
  ic2=ic1++;
  ic2++.retcount(); // considered as ic2.++
 }
```

```
#include <iostream.h>
// page 12-11
// 單一運算元:無參數,有 return值
class incount
{  private:
     int c1,c2;
   public:
     incount()
     { c1=0; c2=1000; }
     incount(int vc1, int vc2) //overloading
     { c1=vc1; c2=vc2; }
     void retcount(void)
     { cout << "c1=" << c1 << endl;
       cout << "c2=" << c2 << endl;}
     incount operator ++()
     { c1++; c2++;
       // unnamed object initialized return
       return incount(c1,c2);
      }
};
void main()
{ incount ic1,ic2;
  ic1.retcount(); ic2.retcount();
  ic1++; ic1.retcount();
  ic2=ic1++;
  ic2++.retcount();
 }
```

```
Ans:c1=0  c2=1000
      c1=0  c2=1000
      c1=1  c2=1001
      c1=3  c2=1003
```

```
Ans:c1=0  c2=1000
      c1=0  c2=1000
      c1=1  c2=1001
      c1=3  c2=1003
```


- 物件相加多元運算 +
- 欲以運算子定義及多元運算設計使物件可直接相加

```
#include <iostream.h>
class room  //Page:12-14
{ private:
    float  ledge,sedge;
  public:
    room() {ledge=0.0; sedge=0.0; }
    room(float le, float se)
      { ledge=le; sedge=se; }
    void getlength()
      { cout << "Input large edge:";
               cin >> ledge;
               cout << "Input small edge:";
               cin >> sedge; }
    void showsquare()
      { cout << (ledge+sedge)*2 << endl;}
    room operator + (room p2);
};
```

```
room room::operator + (room p2)
  { float led=ledge+p2.ledge;
    float sed=sedge+p2.sedge;
    return room(led, sed); }
void main()
{ room r2;
  room r1(3,2);
  r2.getlength();
  cout << "Length of r1 room is:";
  r1.showsquare();
  cout << "Length of r2 room is:";
  r2.showsquare();
  room r3=r1+r2;
  cout << "Length of r3 room is:";
  r3.showsquare();
  room r4=r1+r2+r3;
  cout << "Length of r4 room is:";
  r4.showsquare();
}
```

☐ 物件相加多元運算 +

```
#include <iostream.h> //Page:12-18
#include <math.h>
#include <iomanip.h>
const PI=3.14159;
class rectanglar
{ private:
   double x,y;
   double getr()
     {return sqrt(x*x+y*y);}
   double getangle()
     {return atan(y/x)*180/PI;}
  public:
   rectanglar()
     {x=0;y=0;}
   rectanglar(double p, double q)
     {x=p; y=q;}
   void display1()
     { cout << "(" << x << "," << y << ")=Polar";
       cout << "(" << setw(5) << setprecision(2)
            << getr() << "," << getangle() << ")"; }
   rectanglar operator + (rectanglar r2)
     { double p=x+r2.x;
               double q=y+r2.y;
               return rectanglar(p,q); }
};
```

```
void main()
{ rectanglar r1(20,10);
  rectanglar r2(15,20);
  rectanglar r3=r1+r2;  //considered as r1.+(r2)
  rectanglar r4=r1+r2+r3; // as r1.+(r2.+(r3))
  cout << "\n rectanglar r1";r1.display1();
  cout << "\n rectanglar r2";r2.display1();
  cout << "\n rectanglar r3";r3.display1();
  cout << "\n rectanglar r4";r4.display1();
}
```

Rectanglar (p,q)

r1

r2

r3

r4

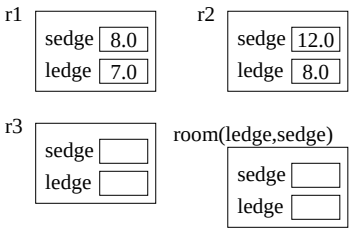

```
#include <iostream.h>
class Distance
{ private:
   int feet;
   float inches;
  public:
   Distance()
     { feet=0; inches=0; }
   Distance(int ft, float in)
     { feet=ft; inches=in; }
   void getdist()
     { cout << "\nEnter feet:";
       cin >> feet;
       cout << "\nEnter inches:";
       cin >> inches; }
   void showdist()
     { cout << feet << "\'-" << inches << '\"'; }
   Distance operator + (Distance);
};
```

```
Distance Distance:: operator + (Distance d2)
{ int f=feet+d2.feet;
  float i=inches+d2.inches;
  if (i>=12.0)
    { i-=12.0;
      f++; }
  return Distance(f,i);
}
void main()
{ Distance dist1,dist3,dist4;
  dist1.getdist();
  Distance dist2(11,6.25);
  dist3=dist1+dist2;
  dist4=dist1+dist2+dist3;
  cout << "\ndist1=";dist1.showdist();
  cout << "\ndist2=";dist2.showdist();
  cout << "\ndist3=";dist3.showdist();
  cout << "\ndist4=";dist4.showdist();
}
```

☐ 物件相加多元運算 :=

```
#include <iostream.h>
class room //Page:12-20
{ private:
   float ledge,sedge;
 public:
   room()
     {ledge=0.0; sedge=0.0;}
   room(float le,float se)
     {ledge=le; sedge=se;}
   void getlength()
     { cout << "Input large edge:"; cin >> ledge;
       cout << "Input small edge:"; cin >> sedge;}
   void showsquare()
     { cout << (ledge+sedge)*2 << endl;}
   room operator += (room p2);
};
room room::operator += (room p2)
{ ledge +=p2.ledge;
 sedge +=p2.sedge;
 return room(ledge, sedge);}
```

```
void main()
{ room r2;
  room r1(8,7);
  r2.getlength();
  cout << "Length of r1 is:";r1.showsquare();
  cout << "Length of r2 is:";r2.showsquare();
  room r3=r1+=r2;
  cout << "Length of r3 is:";r3.showsquare();
  room r4=r1+=r3;
  cout << "Length of r4 is:";r4.showsquare();
}
```



☐ 物件相加多元運算：字串相加

```
#include <iostream.h>
#include <string.h>
const int size=80;
class strings //Page:12-24
{ private:
    char str[size];
  public:
    strings()
      { str[0]='\0';}
    strings(char st[])
      { strcpy(str,st);}
    void printstr()
      { cout << str;}
    strings operator + (strings p);
};
```

```
strings strings::operator + (strings p)
{ if (strlen(str)+strlen(p.str)<size)
     { strings ptemp;
       strcpy(ptemp.str,str);
       strcpy(ptemp.str,p.str);
       return ptemp; }
  else cout << "\n string limited 80 chars"; }

void main()
{ strings ps1="\n London bridge ";
  strings ps2="is falling down !";
  ps1.printstr(); cout << endl;
  ps2.printstr();
  strings ps3=ps1+ps2;
  ps3.printstr();
}
```