Bài thực hành số 2

Mục đích:

- Hàm tạo (Constructors)
- Hàm huỷ (Destructors)
- Định nghĩa chồng các hàm hàm (Function overloading)
- Hàm bạn (Friend functions)

1. Constructor, destructor

```
#include <iostream>
using namespace std;
class CRectangle {
        int nHeight;        //chieu dai hinh chu nhat
        int nWidth;         //chieu rong hinh chu nhat
    public:
        CRectangle();       //ham constructor
        int Area();         //ham tinh dien tich hinh chu nhat
        void Init(int, int);
        ~CRectangle();      //ham destructor
};

CRectangle::CRectangle() {
    cout << "Khoi tao cac thuoc tinh bang ham constructor\n";
    nHeight = 6;
    nWidth = 8;
}
int CRectangle::Area() {
    return nHeight*nWidth;
}

void CRectangle::Init(int Rong,int Dai) {
    cout << "khoi tao lai hcn voi cac bien duoc dua vao\n";
    nWidth = Rong;
    nHeight = Dai;
}
CRectangle::~CRectangle() {
    cout << "destructor dang duoc goi\n";
}

int main()
{

    CRectangle Box,Square;
    cout << "Dien tich cua Box la: "<< Box.Area() << "\n";
    cout << "Dien tich cua Square la: "<< Square.Area() << "\n";
    Box.Init (12,8);
    Square.Init (8,8);
    cout << "Dien tich cua Box la: "<< Box.Area() << "\n";
    cout << "Dien tich cua Square la: "<< Square.Area() << "\n";
}
```

Khi chạy chương trình cho kết quả sau:

```
Khoi tao cac thuoc tinh bang ham constructor
Khoi tao cac thuoc tinh bang ham constructor
Dien tich cua Box la: 48
Dien tich cua Square la: 48
khoi tao lai hcn voi cac bien duoc dua vao
khoi tao lai hcn voi cac bien duoc dua vao
Dien tich cua Box la: 96
Dien tich cua Square la: 64
destructor dang duoc goi
destructor dang duoc goi
```

Các bạn tạo file và gõ tiếp chương trình sau:

```
#include <iostream>
using namespace std;
class CTime {
        int nHours,nMinutes,nSeconds;
   public:
        CTime();   //ham constructor khong doi so
        CTime(int h, int m, int s); //ham constructor 3 doi so
        void Display();
        void AddIt(CTime Time1, CTime Time2);
        ~CTime();  //ham destructor
};
CTime::CTime() {
   cout << "ham constructor mac dinh\n";
   nHours = nMinutes = nSeconds = 0;
}
CTime::CTime(int h, int m, int s) {
   cout << "ham constructor voi 3 doi so\n";
   nHours = h;
   nMinutes = m;
   nSeconds = s;
}

void CTime::Display() {
   cout << nHours << ':' << nMinutes << ':' << nSeconds << endl;
}
void CTime::AddIt(CTime Time1, CTime Time2) {
   nHours = Time1.nHours + Time2.nHours;
   nMinutes = Time1.nMinutes + Time2.nMinutes;
   nSeconds = Time1.nSeconds + Time2.nSeconds;
   if(nSeconds >= 60) {
        nSeconds -= 60;
        nMinutes ++;
   }
   //kiem tra nMinutes < 60
   if(nMinutes >= 60) {
        nMinutes -= 60;
        nHours ++;
   }
}
```

```
CTime::~CTime() {
   cout << "Ham destructor\n";
}

int main() {
   CTime Result;
   CTime T1(1,49,50);
   CTime T2(3,40,30);
   Result.AddIt(T1,T2);
   cout << "Ket qua la: ";
   Result.Display();
}
```

Kết quả sau khi thực hiện chương trình:

```
ham constructor mac dinh
ham constructor voi 3 doi so
ham constructor voi 3 doi so
Ham destructor
Ham destructor
Ket qua la: 5:30:20
Ham destructor
Ham destructor
Ham destructor
```

2. Constructor, destructor và cấp phát bộ nhớ

Tạo file và gõ chương trình sau:

```
#include <iostream>
#include <string.h>
using namespace std;
class Cstring {
   char* strName;
   public:
   CString(char* str){
         int size = strlen(str);
         strName = new char[size + 1];
         strcpy(strName,str);
   }
   void Display(){
         cout << strName << endl;
          cout<<"Chieu dai cua chuoi = "<<strlen(strName)<< endl;
   }
   ~CString(){ delete [] strName; }
};

int main()
{
   CString strFirst("Chao cac ban");
   CString strSecond("Chao tam biet");
   strFirst.Display();
   strSecond.Display();
}
```

Kết quả sau khi thực hiện chương trình:

Chao cac ban
Chieu dai cua chuoi = 12
Chao tam biet
Chieu dai cua chuoi = 13

3. Định nghĩa chồng các hàm

Tạo file và gõ chương trình sau:

```
#include <iostream>
#include <string.h>
using namespace std;
class Cwords  {
        int nNum;
        char strTitle[20];
   public:
        CWords() {
           nNum = 0;
           strcpy(strTitle," ");
        }
        void Display(int i){
             nNum = i;
             cout << "Tham so la " << nNum << endl;
             cout << "Day khong phai la chuoi" << endl;
        }
        void Display(char c){
             strTitle[0] = c;
             cout << "Tham so la " << strTitle[0] << endl;
             cout << "Day la mot ky tu" << endl;
        }
        void Display(char* str){
             strcpy(strTitle,str);
             cout << "Tham so la " << strTitle << endl;
             cout << "Day la mot chuoi" << endl;
        }
        void Display(char* str,int i){
            strcpy(strTitle,str);
                 nNum = i;
            cout<<"Tham so la "<<strTitle<<"and"<<nNum<<endl;
            cout << "Day la mot chuoi" << endl;
        }
};

int main()
{
   CWords X;
   X.Display(120);
   X.Display('A');
   X.Display("String");
   X.Display("String",10);
}
```

Kết quả sau khi thực hiện:

```
Tham so la 120
Day khong phai la chuoi
Tham so la A
Day la mot ky tu
Tham so la String
Day la mot chuoi
Tham so la Stringand 10
Day la mot chuoi
```

4. Hàm bạn (friend function)

```
#include <iostream >
using namespace std;
class B;
class A
{
  float X;
public:
  A() { X = 5.0; }
  friend float Init(A,B);
};

class B {
  float Y;
public:
      B() { Y = 1.0; }
      friend float Init(A,B);
};

float Init(A a,B b) { return a.X + b.Y; }

int main()
{
      A a;
      B b;
      cout << Init(a,b) << " la ket qua cua hai lop\n";
}
```

Kết quả sau khi thực hiện chương trình:

```
6 la ket qua cua hai lop
```

Bài thực hành số 2 (tiếp)

Mục tiêu bài học:

- Định nghĩa chồng các toán tử (operator overloading).
- Định nghĩa chồng toán tử gán.
- Hàm tạo sao chép.
- Chuyển đổi kiểu chuẩn với kiểu do người dùng định nghĩa.

1. Định nghĩa chồng toán tử

Các toán tử ++, --, <= , > +=, có thể định nghĩa chồng cho các kiểu dữ liệu người dùng định nghĩa như các lớp đối tượng. Các toán tử một ngôi và toán tử hai ngôi có thể đa năng hoá như sau:

**** Toán tử một ngôi***

Chương trình sau sẽ định nghĩa chồng các toán tử ++ và --. Toán tử ++ chuyển các ký tự trong chuỗi ký tự thành các ký tự hoa, toán tử -- chuyển các ký tự trong chuỗi ký tự thành các ký tự thường.

- Tạo file mới
- Gõ đoạn chương trình sau:

```
#include <iostream>
#include <string.h>
using namespace std;
class Converter {
           char str[80];
     public:
           Converter()  { strcpy(str,"");   }
           Converter(char *s) { strcpy(str,s);    }
           void display() { cout <<str<< endl;    }
           Converter operator ++()
           {
                 return (strupr(str));
           }
           Converter operator ++(int) {
                 Converter ss = str;
                 char *ptr = strupr(str);
                 strcpy (str, ptr);
                 return ss;
           }
           Converter operator -- ()
           {
                 return (strlwr(str));
           }
           Converter operator --(int) {
                 Converter ss = str;
                 char *ptr = strlwr(str);
                 strcpy(str,ptr);
                 return ss;
           }
};
```

```
int main()
{
            Converter s1 = "changed to UPPERCASE";
            Converter s2 = "BACK TO LOWER CASE";
            Converter s3 = "that is all for now";
            Converter s4 = "ENDING ON A LOW NOTE";
            Converter s5;

            int  i, j;
            ++s1;
            cout << "++s1 = ";
            s1.display();
            --s2;
            cout<<"--s2 = ";
            s2.display();
            s5 = s3++;
            cout << "Result of  s5 = s3++ "<< endl << "s3 = ";
            s3.display();
            cout<<"s5 = ";
            s5.display();
            s4--;
            cout<<"s4-- = ";
            s4.display();
}
```

- Lưu file với tên "danag411.cpp"
- Dịch và chạy chương trình

Kết quả:

```
++s1 = CHANGED TO UPPERCASE
--s2   = back to lower case
Result of  s5 = s3++
s3 = THAT IS ALL FOR NOW
s5 = that is all for now
s4-- = ending on a low note
```

Trong chương trình trên chúng ta xây dựng lớp Converter có một thành phần dữ liệu là mảng ký tự. Chúng ta khai báo file string.h để sử dụng các hàm chuẩn về chuỗi.

Chúng ta sử dụng các hàm chuẩn về chuỗi: strupr() để chuyển chuỗi ký tự thành chuỗi ký tự hoa, strlwr() để chuyển các chuỗi ký tự thành các chuỗi ký tự thường.

Để trình biên dịch phân biệt được các toán tử ++, -- đứng trước hay là đứng sau, chúng ta sử dụng cú pháp sau:

```
    operator++( );        // prefix operation
```

xác định toán tử đứng trước

```
    operator++(int);      // postfix operation
```

xác định toán tử đứng sau.
Tham số int chỉ là giả để trình biên dịch phân biệt hai hình thức trên.

**** Toán tử hai ngôi***

Khi đa năng hoá toán tử hai ngôi, thành phần bên phải toán tử là tham số, thành phần bên trái toán tử là đối tượng thực hiện hàm.

Trong chương trình dưới đây, chúng ta sẽ đa năng hoá toán tử +=. Toán tử này kết hợp phép gán và phép cộng. Trong lệnh sau s3 = s1 += s2, s3 bằng s1 và có kết quả là s1 cộng với s2.

- Tạo file mới
- Gõ đoạn chương trình sau:

```
#include <iostream>
#include <string.h>
define SIZE = 80
using namespace std;
class Cphrase {
            char str[SIZE];
      public:
            CPhrase() {
                  strcpy(str,"  ");
            }

            CPhrase(char *s)  {
                  strcpy(str,s);
            }

            void display()   { cout<<str<<endl; }

            CPhrase operator +=(CPhrase aa) {
                  if (strlen(str) + strlen(aa.str)<SIZE) {
                        strcat(str, aa.str);
                        return (*this);
                  }
                  else {
                        cout<<"string is too long";
                        return (*this);
                  }
            }
};
int main()
{
            CPhrase s1;
            CPhrase s2 = " Chao cac ban";
            s1+= s2;
            s1.display();
            s1 = " Chuc suc khoe";
            CPhrase s3;
            s3 = s1 += s2;
            s3.display();
            CPhrase s4;
            CPhrase s5 = " Xin moi vao";
            s4 = s2 += s5 += s5;
            s4.display();
}
```

- Lưu file với tên "danag412.cpp"
- Dịch và chạy chương trình

Kết quả:

```
Chao cac ban
Chuc suc khoe Chao cac ban
Chao cac ban Xin moi vao Xin moi vao
```

Nếu bạn chỉ sử dụng lệnh đơn giản là s1 += s2 thì trong hàm không cần trả về giá trị. Tuy nhiên để có được lệnh phức tạp hơn như s3 = s1 +=s2, s4 = s2 += s5 += s5 phải có kiểu trả về.

2. Định nghĩa chồng toán tử gán và khởi tạo chép đối tượng

Xây dựng lớp CString như sau:

- Tạo file mới và gõ đoạn chương trình sau:

```
#include <iostream>
#include <string.h>
using namespace std;
class CString  {
            char* str;
      public:
            CString (char* s="")  {
                  int size = strlen(s);
                  str = new char[size+1];
                  strcpy(str, s);
            }
            CString (CString &ss) {
                  str = new char[strlen(ss.str) + 1];
                  strcpy(str, ss.str);
            }
            ~CString() { delete str; }
            CString& operator = (CString& ss) {
                  delete str;
                  str = new char[strlen(ss.str) + 1];
                  strcpy(str, ss.str);
                  return *this;
            }
            void showstring() { cout << str<<endl; }
};
int main() {
      CString s1 = " Chao cac ban";
      cout <<"s1 = ";
      s1.showstring();
      CString s2(s1), s3;
      s3 = s1;
      cout << "s2 = ";
      s2.showstring();
      cout << "s3 = ";
      s3.showstring();
      CString s4 = s1;
      cout << "s4 = ";
      s4.showstring();
}
```

- Lưu file với tên "con42.cpp"
- Dịch và chạy chương trình

Kết quả:

```
s1 =  Chao cac ban
s2 =  Chao cac ban
s3 =  Chao cac ban
s4 =  Chao cac ban
```

3. Chuyển đổi kiểu chuẩn với kiểu người dùng định nghĩa

Chuyển đổi kiểu qua lại giữa kiểu cơ bản và kiểu đối tượng.

- Tạo file mới
- Gõ đoạn chương trình sau:

```
#include <iostream>
#include <string.h>
define SIZE = 80
using namespace std;
class CString  {
           char str[SIZE];
      public:
           CString () { strcpy(str,"  ");   }
           CString (char *s)  {strcpy (str,s); }
           void getstr()
           {
                cout<< "Nhap vao chuoi: ";
                cin >>str;
           }

           void display()
           {
                cout <<str<<endl;
           }
           operator char*()
           {
                return (str);
           }
};
int main(){
   CString s1;
   char* p;
   s1.getstr();
   s1.display();
   p = s1;
   cout << "Chuoi p = "<<p <<endl;
   char n[15] = "Chuoi  moi";
   s1 = n;
   s1.display();
}
```

- Lưu file với tên "con42.cpp"
- Dịch và chạy chương trình

Kết quả:

```
Nhap vao chuoi: hello
hello
Chuoi p = hello
Chuoi  moi
```

Trong hàm main(), câu lệnh sau:

p = s1;

gán cho con trỏ kiểu ký tự p đối tượng s1. Khi trình biên dịch thấy vế phải của phép gán là đối tượng và vế trái là con trỏ ký tự thì nó sử dụng hàm chuyển kiểu.

s1 = n;

lệnh này trình biên dịch tự động gọi hàm khởi tạo có một tham số (vì s1 là đối tượng của lớp Cstring), nếu hàm khởi tạo này không được định nghĩa thì trình biên dịch sẽ báo lỗi.

Bài tập làm thêm

TH2.1: Viết một lớp Hình chữ nhật, gồm có các thuộc tính (private) là cạnh dài, rộng; các phương thức (public): nhập, tính diện tích của hình chữ nhật. Viết hàm main() để: Nhập một dãy n hình chữ nhật, tìm hình có diện tích lớn nhất.

TH2.2: Viết một lớp Hình chữ nhật, gồm có các thuộc tính (private) là cạnh dài, rộng; các phương thức (public): nhập, tính diện tích của hình chữ nhật, xác định một hình chữ nhật có phải hình vuông không. Viết hàm main() để: Nhập một dãy n hình chữ nhật, tìm hình vuông có diện tích lớn nhất.

TH2.3: Xây dựng lớp Đa thức, trong đó định nghĩa: các hàm tạo có đối, và định nghĩa chồng các phương thức toán tử <<, >>, +, =.

TH2.4: Xây dựng lớp Vector với thành các hàm tạo, hàm tạo sao chép, hàm hủy, định nghĩa chồng các toán tử gán =, toán tử chỉ số [], toán tử so sánh ==, toán tử nhập >>, toán tử xuất <<.

```
class Vector
{
     int size;
     float *data;
public:
     Vector(int =1, float = 0.0);
     Vector(const Vector&);
     ~Vector();
     float operator[](int);
     ...
};
```