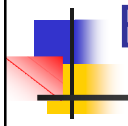
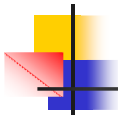


Bài 9: Mẫu(Template)



Nội dung

- Định nghĩa và cách sử dụng template
- Cách tạo một lớp template
- Cách tạo một hàm template

Hàm hoán đổi vị trí của 2 số

```
void Hoanvi_int(int &x, int &y){
    int temp = x;
    x = y;
    y = temp;
}
void Hoanvi_float(float &x, float &y){
    float temp  = x;
    x = y;
    y = temp;
}
```

Khai báo hàm mẫu

Cú pháp:

template < class Tên_mẫu>

Khi đó ta có thể sử dụng Tên_mẫu như một kiểu dữ liệu để làm đối số hoặc kiểu trả về cho một hàm nào đó.

Ví dụ khai báo một hàm với template

template <class C>

C function(C param);

Định nghĩa hàm với template

```
template <class C>
C  function( C  param){
  …
}
```

Hàm hoán vị với template

Khai báo nguyên mẫu hàm

```
template <class T>
void hoanvi(T &x, T&y);
```

Định nghĩa hàm

```
template <class T>
void hoanvi(T &x, T&y){
    T temp;
    temp = x;
    x = y;
    y = temp;
}
```

Hàm sắp xếp với template

```
template <class T>
void sapxep(T a[], int n);
int main()
{
    char c[100];          int a[100];
    float b[100];         int i, n =20;
    for(i = 0; i<n; i++)
    {
        c[i] = rand() % 26 +65;
        a[i] = rand();
        b[i] = rand() /99.0;
    }
```

Hàm sắp xếp với template(tiếp)

```
    sapxep(c, n);
    sapxep(a, n);
    sapxep(b, n);
    for(i= 0; i< n; i++)
    {
        cout.width(10);              cout<<c[i];
        cout.width(10);              cout<<a[i];
        cout.setf(ios::fixed);       cout.precision(6);
        cout.width(20);              cout<<b[i];
        cout<<endl;
    }
return 0;}
```

Hàm sắp xếp với template(tiếp)

```
template <class T>
void sapxep(T a[], int n)
{
    int i, j;
    for(i =0; i< n-1; i++)
        for(j =i+1; j<n; j++)
            if(a[i] > a[j])
            {
                T temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
}
```

Khai báo một lớp với template

```
template <class T>
class Array{
public:
    …
    Array(const Array<T> & init)
…
private:
    T *pType;
    …
};
```

Khai báo phương thức có tham số là kiểu mẫu

Cú pháp:

```
        void method(tên_mẫu tên_tham_số);
```

Ví dụ:

```
template <class T >
class Point
{
    void SetX(const T &x);
private:
    T  v_x, v_y;
};
```

Định nghĩa phương thức

Định nghĩa hàm:

```
template <class T>
void Point<T>::SetX(const T &x)
{
    v_x = x;
}
```

Chú ý:

- Point<T> thay cho Point
- Luôn có dòng khai báo template trước mỗi định nghĩa hàm

Hàm có kiểu trả về là kiểu mẫu

Cú pháp:

T method(các_tham_số);

Ví dụ:

```
template <class T >
class Point
{
   T GetX()const;
private:
   T  v_x, v_y;
};
```

Hàm có kiểu trả về là kiểu mẫu

Định nghĩa hàm:

```
template <class T>
T   Point<T>::Getx()const
{
    return v_x;
}
```

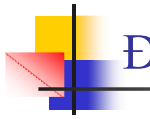
Đối số của hàm là đối tượng của lớp

Cú pháp:

Kiểu method(Tên_lớp<T> param);

Ví dụ:

```
template <class T >
class Point
{
   Point(const Point<T> &a);
private:
   T  v_x, v_y;
};
```



Đối số của hàm là đối tượng của lớp

```
template <class T>
Point< T>::Point(const Point<T> &a)
{
    v_x = a.GetX();
    v_y = a.GetY();
}
```

Sử dụng template

Với cách khai báo lớp mẫu Array trên ta có thể tạo ra lớp mảng với nhiều kiểu dữ liệu khác nhau

```
Array<int> theIntArray;
Array<float> thefloatArray;
Array<CAT> theCatArray;
Array<Vector> theIntArray;
```

…

Với lớp Point ta có thể sử dụng như sau:

```
Point <float> p1, p2;
Point <int> q1, q2;
```