Lập trình hướng đối tượng và C++

Bài 9: Khuôn hình

TS. Nguyễn Hiếu Cường

Bộ môn CNPM, Khoa CNTT, Trường ĐH GTVT

Email: cuonggt@gmail.com

Hà Nội, 2022

Nội dung chính

1. Giới thiệu môn học
2. Các khái niệm cơ bản
3. Hàm trong C++
4. Lớp và đối tượng
5. Định nghĩa chồng toán tử
6. Hàm tạo và hàm huỷ
7. Dẫn xuất và thừa kế
8. Tương ứng bội
9. **Khuôn hình**

Khuôn hình hàm

- Định nghĩa chồng hàm (function overloading)
  - Cùng tên hàm, có các đối khác nhau
  - Các hàm phân biệt thông qua các đối khác nhau
- Định nghĩa lại hàm (function overriding)
  - Cùng tên hàm, cùng danh sách đối
  - Các hàm (virtual function) được thực hiện tùy theo đối tượng
- Khuôn hình hàm (function template)
  - Định nghĩa khuôn mẫu cho một họ các hàm (cùng tên hàm và đối)
  - Hàm được xây dựng với những kiểu tham số trừu tượng, khi có lời gọi đến hàm, chương trình dịch sẽ tự động tạo ra những biến thể thích hợp nhất của hàm

Khai báo và xây dựng khuôn hình hàm

- Khai báo:

  template <class T_1, class T_2, …, class T_n>

  Nội dung hàm

- Các hàm được xây dựng tương tự như các hàm bình thường, nhưng các kiểu dữ liệu cụ thể (int, float…) được thay bằng các khuôn hình (T_1,…, T_n)

- Ví dụ:

```
template <class T>
T max2 (T x, T y) { return (x>y)?x:y; }
```

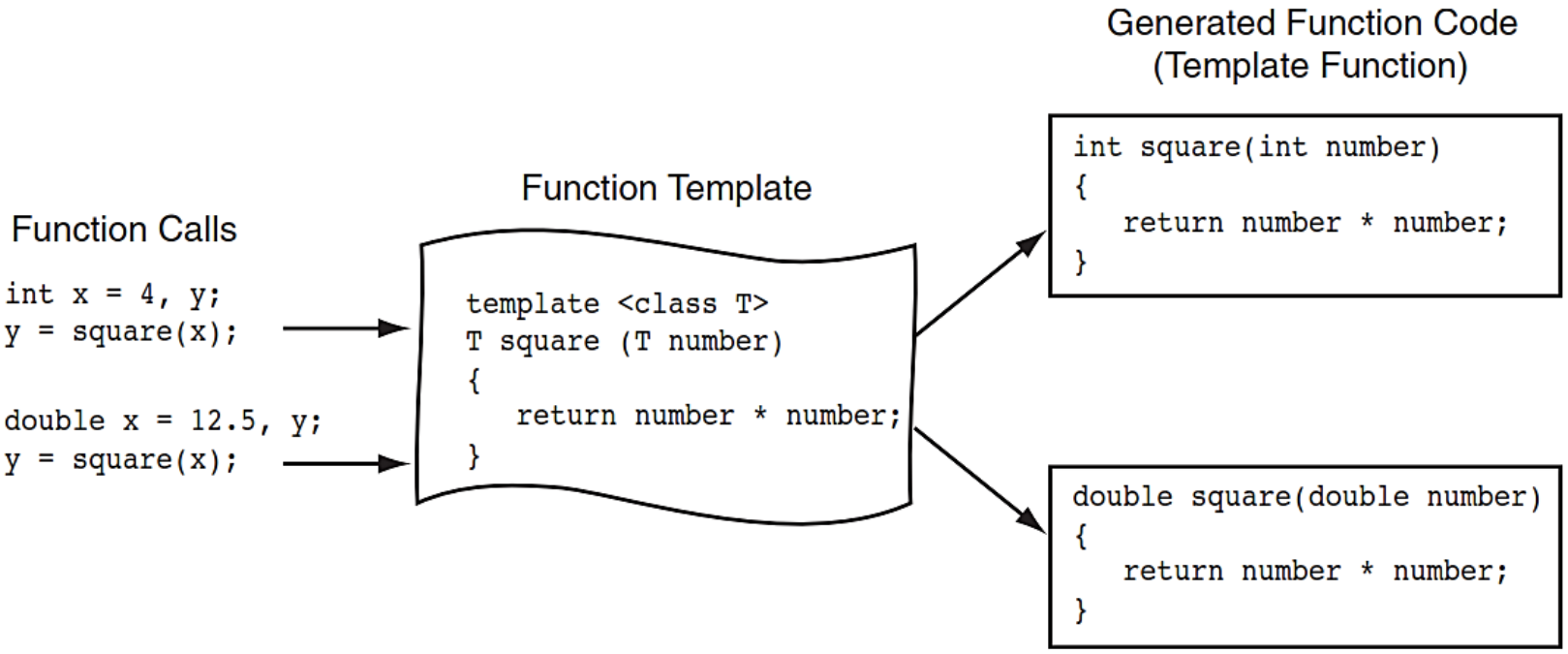
Sử dụng khuôn hình hàm

- Khi gọi hàm trình biên dịch (compiler) sẽ sinh hàm tương ứng với kiểu dữ liệu trong lời gọi
- Ví dụ: Sử dụng khuôn hình hàm max2

```
int a, b, c;
float x, y, z;
PS u, v, t;
...
c = max2(a,b);
z = max2(x,y);
t = max2(u,v);
```

```
#include <iostream>

using namespace std;

// Template definition for square function.
template <class T>
T square(T number)
{
     return number * number;
}

int main()
{
     int userInt;        // To hold integer input
     double userDouble; // To hold double input


     cout << "Enter an integer and a floating-point value: ";
     cin >> userInt >> userDouble;
     cout << "Here are their squares: ";
     cout << square(userInt) << " and "
          << square(userDouble) << endl;

}
```

Thực chất khi gọi hàm



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

template <class T1, class T2>
int largest(const T1 &var1, T2 &var2)
{
    if (sizeof(var1) > sizeof(var2))
        return sizeof(var1);
    else
        return sizeof(var2);
}

int main()
{
    int i = 0;
    char c = ' ';
    float f = 0.0;
    double d = 0.0;

    cout << "Comparing an int and a double, the largest\n"
         << "of the two is " << largest(i, d) << " bytes.\n";

    cout << "Comparing a char and a float, the largest\n"
         << "of the two is " << largest(c, f) << " bytes.\n";

    return 0;
}
```

Ví dụ (khuôn hình hàm sắp xếp)

```
template <class T>
void sapxep(T *x, int n) {
  int i, j;  T tg;
  for(i=1; i<=n-1; ++i)
    for(j=1; j<=n; ++j) {
      if(x[j] > x[i]) { tg=x[i]; x[i]=x[j]; x[j]=tg; }
    }
}
int main() {
  int a[100];
  float x[100];
  ...
  sapxep(a, n);
  sapxep(x, m);
}
```

Kinh nghiệm thực tế

- Bước 1: Xây dựng như hàm thông thường với đối là một kiểu nào đó
  - Ví dụ swapVars dùng để hoán vị hai số

```
void swapVars(int &var1, int &var2)
{
     int temp;

     temp = var1;
     var1 = var2;
     var2 = temp;
}
```

- Bước 2: Sau khi đã kiểm thử tính đúng đắn của hàm trên, ta thêm

```
template<class T>
```

và thay các kiểu cụ thể bằng **T**

Ví dụ

- Xây dựng khuôn hình hàm tính trung bình cộng một dãy số. Trong hàm main() hãy thực hiện hàm với các dãy kiểu int, long, double, char.

```
template <class atype>              //function template
atype avg(atype* array, int size)
   {
   atype total = 0;
   for(int j=0; j<size; j++)        //average the array
      total += array[j];
   return (atype)total/size;
   }
//////////////////////////////////////////////////////////////
int intArray[] =       {1, 3, 5, 9, 11, 13};
long longArray[] =     {1, 3, 5, 9, 11, 13};
double doubleArray[] = {1.0, 3.0, 5.0, 9.0, 11.0, 13.0};
char charArray[] =     {1, 3, 5, 9, 11, 13};

int main()
   {
   cout << "\navg(intArray)=" << avg(intArray, 6);
   cout << "\navg(longArray)=" << avg(longArray, 6);
   cout << "\navg(doubleArray)=" << avg(doubleArray, 6);
   cout << "\navg(charArray)=" << (int)avg(charArray, 6) << endl;
   return 0;
   }
```

Khuôn hình lớp

- Khuôn hình lớp tạo ra một họ các lớp giống nhau về bản chất xử lý, nhưng khác nhau về kiểu dữ liệu
- Trong khuôn hình lớp các thành phần dữ liệu hoặc các hàm thành phần của lớp sử dụng các kiểu dữ liệu trừu tượng
- Khi chương trình có nhu cầu sử dụng lớp với những kiểu dữ liệu cụ thể thì chương trình dịch sẽ tự động biên dịch những thể hiện thích hợp nhất

Khai báo và định nghĩa khuôn hình lớp

- Khai báo:

template <class T_1, class T_2, …, class T_n>

Nội dung lớp

- Các lớp được xây dựng tương tự như các lớp bình thường, chỉ khác các kiểu dữ liệu cụ thể được thay bằng các khuôn hình (T_1,…, T_n)

Ví dụ

```
template<class T, int n>
class array
{
    T elem[n];
public:
    T& operator[](int i)
    {
        return elem[i];
    }
};
```

```
int main()
{
    array<int,10> a;
    for(int i=0; i<10; ++i)
    {
        a[i]= i*i;
        cout<<a[i]<<"  ";
    }
    array<float,20> b;
    for(int i=0; i<20; ++i)
    {
        b[i]= i*i*1.5;
        cout<<b[i]<<"  ";
    }
}
```

Ví dụ

```
#include <iostream>
using namespace std;
template <class T>
class mypair {
    T a, b;
  public:
    mypair (T first, T second)
      { a= first; b= second; }
    T getmax ();
};
template <class T>
T mypair<T>::getmax () {
  T retval;
  retval = a>b? a : b;
  return retval;
}
```

```
int main ()
{
  // Sử dụng
  mypair <int> myobject(100, 75);
  cout << myobject.getmax();
}
```

Tóm tắt

- Khuôn hình hàm (function template)
- Khuôn hình lớp (class template)
  - Ý nghĩa của khuôn hình (mẫu)
  - Cách xây dựng các khuôn hình
  - Sử dụng các khuôn hình với các kiểu dữ liệu cụ thể

Câu hỏi

1. A template provides a convenient way to make a family of
   a. variables.
   b. functions.
   c. classes.
   d. programs.
2. A template argument is preceded by the keyword _____.
3. True or false: Templates automatically create different versions of a function, depending on user input.
4. Write a template for a function that always returns its argument times 2.
5. A template class
   a. is designed to be stored in different containers.
   b. works with different data types.
   c. generates objects which must all be identical.
   d. generates classes with different numbers of member functions.

Câu hỏi

6. True or false: There can be more than one template argument.
7. Creating an actual function from a template is called _____ the function.
8. Actual code for a template function is generated when

   a. the function declaration appears in the source code.

   b. the function definition appears in the source code.

   c. a call to the function appears in the source code.

   d. the function is executed at runtime.
9. The key concept in the template concept is replacing a _____ with a name that stands for _____.
10. Templates are often used for classes that _____ _____.

Bài tập (khuôn hình hàm)

1. Xây dựng khuôn hình hàm tìm max của 2 số, áp dụng tìm max của 2 số nguyên, 2 số thực, và 2 phân số (phải định nghĩa operator**>**).

2. Xây dựng khuôn hình hàm tính max của một dãy số. Trong hàm main() hãy thực hiện hàm với một số kiểu dữ liệu.

3. Xây dựng khuôn hình hàm hoanvi, áp dụng trong khuôn hình hàm sắp xếp dãy số. Sử dụng các khuôn hàm trên để sắp xếp giá trị hai mảng: một mảng số nguyên và một mảng số thực.

Bài tập (khuôn hình lớp)

4. Xây dựng lớp Stack để chứa các số int.

5. Xây dựng khuôn hình lớp Stack (ngăn xếp).

6. Xây dựng khuôn hình lớp Queue (hàng đợi).

7. Xây dựng khuôn hình lớp SingleLinkedList (danh sách liên kết đơn).

8. Xây dựng lớp Vector