THAM SỐ HÓA KIỂU DỮ LIỆU

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

GVHD: Trương Toàn Thịnh

NỘI DUNG

- Giới thiệu
- Tham số hóa cho hàm & lớp
- Xử lý trường hợp đặc biệt
- Tham số hóa với các thông tin phụ
- Kế thừa các lớp được tham số hóa
- Các vấn đề khác

GIỚI THIỆU

- Xét tình huống hàm swap

```
void swap(int& a, int& b){
  int temp = a;
  a = b;
  b = temp;
}
void swap(float& a, float& b){
  float temp = a;
  a = b;
  b = temp;
}
```

GIỚI THIỆU

- Cần cơ chế tổng quát hóa kiểu dữ liệu
  - Nâng cao tính tái sử dụng mã nguồn
  - Không lặp lại đoạn mã
  - Tập trung phát triển tính năng chương trình
- Tham số hóa cho hàm swap

```
template <class T>
void swap(T& a, T& b){
  T temp = a;
  a = b;
  b = temp;
}
```

NỘI DUNG

- Giới thiệu
- Tham số hóa cho hàm & lớp
- Xử lý trường hợp đặc biệt
- Tham số hóa với các thông tin phụ
- Kế thừa các lớp được tham số hóa
- Các vấn đề khác

THAM SỐ HÓA CHO HÀM & LỚP

- Cú pháp tham số hóa cho hàm

```
template <class T>
<Kiểu dữ liệu> Tên_hàm(T){
  //…
}
template <class T, class U>
<Kiểu dữ liệu> Tên_hàm(T, U){
  //…
}
```

THAM SỐ HÓA CHO HÀM & LỚP

- Cú pháp tham số hóa cho lớp (viết trong cùng một tập tin)

```
template <class T>
class TênLớp{
  //…
};
template <class T>
<Kiểu dữ liệu> TênLớp<T>::TênHàm(…){
  //…
}
```

THAM SỐ HÓA CHO HÀM & LỚP

- Ví dụ

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

template <class T>
class MyArray {
  private:
    T* pArr;
    int n;
  public:
    MyArray();
    ~MyArray();
};
```

```
template <class T>
MyArray<T>::MyArray() {
  n = 0;
  pArr = NULL;
}

template <class T>
MyArray<T>::~MyArray() {
  if (n > 0) {
    delete[] pArr;
    n = 0;
  }
}
```

NỘI DUNG

- Giới thiệu
- Tham số hóa cho hàm & lớp
- Xử lý trường hợp đặc biệt
- Tham số hóa với các thông tin phụ
- Kế thừa các lớp được tham số hóa
- Các vấn đề khác

XỬ LÝ TRƯỜNG HỢP ĐẶC BIỆT

- Khi có nhu cầu xử lý riêng cho một kiểu dữ liệu nào đó, ta có thể viết hàm bổ sung
- Ví dụ hàm template sau

```
template <class T>
void dichchuyen(T& x, int k){x += k;}
```

- Ta có nhu cầu xử lý riêng nếu T là char

```
template <>
void dichchuyen<char>(char& x, int k) {
  if(x == toupper(x)){
    if(x + k > 'Z') x = 'A' + x + k – 'Z' – 1;
    else x += k;
  }
  else{
    if(x + k > 'Z') x = 'a' + x + k – 'z' – 1;
    else x += k;}}
```

XỬ LÝ TRƯỜNG HỢP ĐẶC BIỆT

- Đối với lớp ta cũng có cú pháp tương tự

```
template <>
class Tên_lớp<char>{
  //...
}
```

THAM SỐ HÓA KIỂU DỮ LIỆU VỚI THÔNG TIN PHỤ

- Có thể cung cấp thêm thông tin phụ
- Xét ví dụ lớp MyArray

```
template <class T, int SIZE>
class MyArray{
  public:
    //…
  private:
    T arr[SIZE];
};
```

```
template <class T = int, int SIZE = 100>
class MyArray{
  public:
    //…
  private:
    T arr[SIZE];
};
```

- Trong hàm main ta có thể khai báo

```
MyArray<int, 50> a; //MyArray<> a;
```

NỘI DUNG

- Giới thiệu
- Tham số hóa cho hàm & lớp
- Xử lý trường hợp đặc biệt
- Tham số hóa với các thông tin phụ
- Kế thừa các lớp được tham số hóa
- Các vấn đề khác

KẾ THỪA CÁC LỚP THAM SỐ HÓA

- Xét ví dụ

```
#include <iostream>
using namespace std;
template <class T>
class A{
  public:
    void doSth(){cout << "abc…" << endl}
  private:
    T a;
};
```

```
template <class T>
class B: public A<T>{
  public:
    void doSthElse(){
      doSth();
    }
};
```

- Khi dùng gcc biên dịch sẽ bị lỗi không tìm thấy hàm 'doSth()'. Có thể dùng các cách
  - Sửa thành: this->doSth()
  - Sửa thành: A<T>::doSth()

NỘI DUNG

- Giới thiệu
- Tham số hóa cho hàm & lớp
- Xử lý trường hợp đặc biệt
- Tham số hóa với các thông tin phụ
- Kế thừa các lớp được tham số hóa
- Các vấn đề khác

CÁC VẤN ĐỀ KHÁC

- Xảy ra lỗi liên kết nếu tách phần cài đặt và khai báo ra hai tập tin '.h' và '.cpp'

```
template <class T>
class MyArr{
 private:
   T* arr;
   int n;
 public:
   MyArr();
   ~MyArr();
};
```

```
#include "MyArr.h"
template <class T>
MyArr<T>::MyArr(){
  n = 0;
  arr = NULL;
}
MyArr<T>::~MyArr(){
  if(n > 0){
    delete[] arr;
    n = 0;
    arr = NULL;
  }
}
```

  - Gộp chung để sửa lỗi
  - Thêm dòng lệnh: template class MyArr<int>; ở cuối tập tin MyArr.cpp (Bất tiện vì cần biết trước kiểu sử dụng)

CÁC VẤN ĐỀ KHÁC

- Xảy ra lỗi khi dùng hàm friend
  - Định nghĩa hàm friend trong khai báo lớp

```
template <class T>
class A{
  private:
    int a;
  public:
    friend ostream& operator<<(ostream& o, const A<T>& src){
      o<<"The value is: " << src.a << endl;
      return o;
    }
};
```

  - Dùng kí hiệu '<>' tại hàm friend trong lớp

```
template <class T>
class A{
  private: int a;
  public: friend ostream& operator<<<>(ostream&, const A<T>&);
};
```