CHƯƠNG 2

CÁC MỞ RỘNG CỦA C++

CÁC MỞ RỘNG CỦA NGÔN NGỮ C++

Chương 2 trình bày những vấn đề sau đây:

1. *Giới thiệu chung về ngôn ngữ C++*
2. *Xuất nhập dữ liệu trong C++*
3. *Cấp phát và giải phóng bộ nhớ*
4. *Biến tham chiếu, hằng tham chiếu*
5. *Truyền tham số cho hàm theo tham chiếu*
6. *Hàm trả về giá trị tham chiếu*
7. *Hàm với đối số có giá trị mặc định*
8. *Hàm tải bội* (Hàm quá tải)

2.1. Giới thiệu chung về C++

- C++ là ngôn ngữ lập trình hướng đối tượng và là sự mở rộng của ngôn ngữ C. Vì vậy mọi khái niệm trong C đều dùng được trong C++. Phần lớn các chương trình C đều có thể chạy được trong C++.
- Trong chương 2 chỉ tập trung giới thiệu những khái niệm, đặc tính mới của C++ hỗ trợ cho lập trình hướng đối tượng.
- Một số kiến thức có trong C++ nhưng đã có trong ngôn ngữ C sẽ không được trình bày lại.

2.2. Xuất nhập dữ liệu trong C++

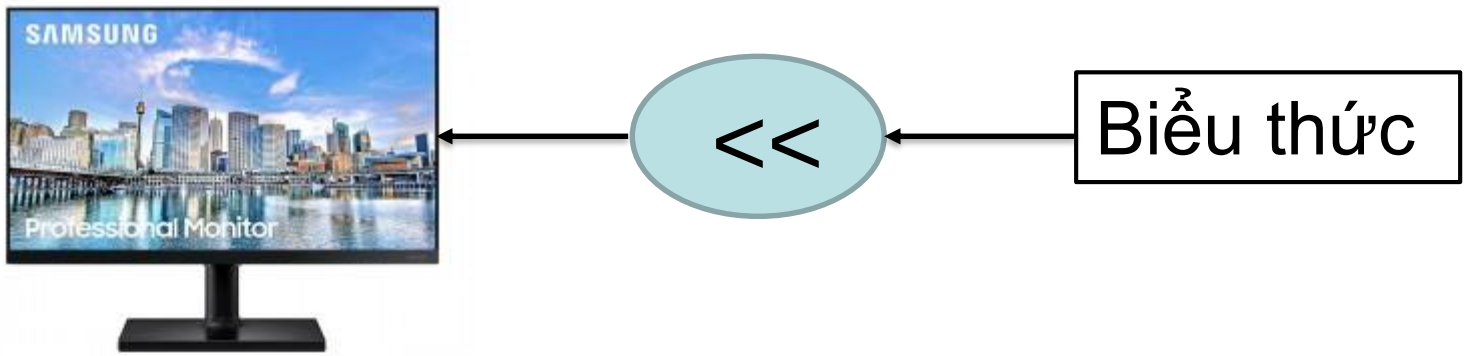
Để xuất dữ liệu ra màn hình và nhập dữ liệu từ bàn phím, trong C++ vẫn có thể dùng hàm printf() và scanf(), ngoài ra trong C++ ta có thể dùng dòng xuất/nhập chuẩn để nhập/xuất dữ liệu thông qua hai biến đối tượng của dòng (stream object) là **cout** và **cin**.

2.2.1. Xuất dữ liệu

cout << biểu thức 1<<. . .<< biểu thức N;

Trong đó **cout** được định nghĩa trước như một đối tượng biểu diễn cho thiết bị xuất chuẩn của C++ là màn hình, **cout** được sử dụng kết hợp với **toán tử chèn** << để hiển thị giá trị các biểu thức 1, 2,..., N ra màn hình.

Minh họa việc xuất biểu thức ra màn hình




```
int a = 1; float b = 2.5;
cout<<a<<b;  //12.5
//cout<< "\n a = “<<a<<endl;  //a = 1
//cout<< "\n b = “<<b<<endl;  //b = 2.5
```

2.2.2. Nhập dữ liệu

cin >>biến 1>>. . . >>biến N;

Toán tử **cin** được định nghĩa trước như một đối tượng biểu diễn cho thiết bị vào chuẩn của C++ là bàn phím, **cin** được sử dụng kết hợp với **toán tử trích** >> để nhập dữ liệu từ bàn phím cho các biến 1, 2, ..., N.


```
int a ; float b;
cout<<"\n a = "; cin>>a;
cout<<"\n b = "; cin>>b;
//cin>>a>>b;
```

Chú ý:

- Để nhập một chuỗi không quá n ký tự và lưu vào mảng một chiều a (kiểu char) có thể dùng hàm cin.get như sau:

  cin.get(a,n);

- Toán tử nhập cin>> sẽ để lại ký tự chuyển dòng '\n' trong bộ đệm. Ký tự này có thể làm trôi phương thức cin.get. Để khắc phục tình trạng trên cần dùng phương thức

  cin.ignore();

  để bỏ qua ký tự chuyển dòng.

- Để sử dụng các biến đối tượng cần khai báo tập tin dẫn hướng **iostream.h** trong C++ hoặc **iostream** trong DEV C++

- Trong C++, có thể khai báo biến chuỗi là một đối tượng của lớp **string**. Để khai báo biến chuỗi ta dùng cú pháp:

  string Danh sách tên biến chuỗi;

  Lúc đó, để nhập một chuỗi từ bàn phím, ta dùng hàm **getline()** theo cú pháp:

  getline(cin, tên biến chuỗi);

Ví dụ:

```
string str;
cout<<"Nhap mot chuỗi ky tu: ";
getline(cin,str);
```

2.2.3. Định dạng khi in ra màn hình

- Để quy định số thực được hiển thị ra màn hình với p chữ số sau dấu chấm thập phân, ta dung lệnh sau,

  cout<<fixed<<setprecision(p);

- Để quy định độ rộng tối thiểu để hiển thị là **k** vị trí cho giá trị (nguyên, thực, chuỗi) ta dùng hàm: setw(k)

  Hàm này cần đặt trong toán tử xuất và nó chỉ có hiệu lực cho một giá trị được in gần nhất. Các giá trị in ra tiếp theo sẽ có độ rộng tối thiểu mặc định là 0, như vậy câu lệnh:

  cout<<setw(6)<<“Khoa”<<“CNTT”

  sẽ in ra chuỗi “ KhoaCNTT”.

  Chú ý: Khai báo tập tin **iomanip** để sử dụng đượccác định dạng ở trên:

  #include <iomanip>

Ví dụ 2.5.

Chương trình sau cho phép nhập một danh sách không quá 100 thí sinh. Dữ liệu mỗi thí sinh gồm họ tên, điểm của 3 môn thi 1, 2, 3 và tổng điểm. Sau đó in danh sách thí sinh theo thứ tự giảm dần của tổng điểm.

```
#include <iostream>
#include <iomanip>
using namespace std;
struct hocsinh
  {
    char ht[25];  //string ht;
    float d1,d2,d3,td;
   };

void nhap(hocsinh ts[], int n)
{ for (int i=0;i<n;++i)
    {
          cin.ignore();
          cout << "\n Thi sinh:"<<i+1;
          cout << "\n Ho ten: ";
          cin.get(ts[i].ht,25);
          cout << "Diem cac mon thi :";
          cin>>ts[i].d1>>ts[i].d2>>ts[i].d3;
          ts[i].td=ts[i].d1+ts[i].d2+ts[i].d3;


    }
```

```
void sapxep(hocsinh ts[], int n)
{   for (int i=0;i<n-1;++i)
      for(int j=i+1;j<n;++j)
        if(ts[i].td<ts[j].td)
                    {
                          hocsinh tam;
                          tam = ts[i];
                          ts[i]=ts[j];
                          ts[j]=tam;
                    }
}
void xuat(hocsinh ts[], int n)
{   cout<< "\n Danh sach sau khi sap xep : "<<endl;
    for (int i=0;i<n;++i)
       cout<<setw(30)<< ts[i].ht
             <<setw(5)<<ts[i].td<<endl;
}
```

```
int main()
{
    hocsinh ts[100];
    int n;
    cout << "Nhap so thi sinh: ";
    cin >> n;  //de lai ky tu chuyen dong trong bo dem
    nhap(ts,n);
    sapxep(ts,n);
    xuat(ts,n);
    return 0;
}
```

2.3. Cấp phát và giải phóng bộ nhớ

Trong C có thể sử dụng các hàm cấp phát bộ nhớ như malloc(), calloc() và hàm free() để giải phóng bộ nhớ được cấp phát. C++ đưa thêm một cách thức mới để thực hiện việc cấp phát và giải phóng bộ nhớ bằng cách dùng hai toán tử **new** và **delete.**

2.3.1. Toán tử new để cấp phát bộ nhớ

Toán tử new có cú pháp như sau:

biến con trỏ = new Tên kiểu ;

Trong đó Tên kiểu là **kiểu dữ liệu của biến con trỏ**, nó có thể là: các kiểu dữ liệu chuẩn như int, float, double, char,... hoặc các kiểu do người lập trình định nghĩa như mảng, cấu trúc, lớp,...

Ví dụ:

```
int *p;
p = new int;  //cấp phát vùng nhớ cho biến con trỏ p
```

Hoặc khai báo và cấp phát:

```
int * p = new int;
```

Chú ý:

Để cấp phát bộ nhớ cho mảng một chiều, ta dùng cú pháp như sau:

Biến con trỏ = new kiểu [n];

Trong đó n là số nguyên dương xác định số phần tử của mảng.

Ví dụ: int *a = new int [100];

/* cấp phát bộ nhớ để lưu trữ mảng một chiều a gồm 100 phần tử */

Thảo luận: Hãy cho biết sự khác nhau của 2 cách khai báo sau:

```
int a[100];
int *a = new int [100];
```

Khi sử dụng toán tử **new** để cấp phát bộ nhớ, nếu không đủ bộ nhớ để cấp phát, **new** sẽ trả lại giá trị NULL cho con trỏ.

Đoạn chương trình sau minh họa cách kiểm tra lỗi cấp phát bộ nhớ:

```
int *p;
int n;
cout<< “\n So phan tu : ”;
cin>>n;
p = new int [n];
if (p == NULL)
    {
     cout << “Loi cap phat bo nho”;
     exit(1);
    }
```

2.3.2. Toán tử delete

Toán tử delete thay cho hàm free() của C, nó có cú pháp như sau:

delete con trỏ ;

Ví dụ 2.6 Chương trình sau minh hoạ cách dùng new để cấp phát bộ nhớ chứa n thí sinh. Mỗi thí sinh là một cấu trúc gồm các trường sobd (số báo danh), ht(họ tên) và td (tổng điểm).

Chương trình sẽ nhập n, cấp phát bộ nhớ chứa n thí sinh, kiểm tra lỗi cấp phát bộ nhớ, nhập n thí sinh, sắp xếp thí sinh theo thứ tự giảm của tổng điểm, in danh sách thí sinh sau khi sắp xếp, giải phóng bộ nhớ đã cấp phát.

Mã lệnh:

```
#include <iomanip>
#include <iostream>
#include <stdlib.h>
using namespace std;

struct TS
{
  char ht[20];  //string ht
  long sobd;
  float td;             //tong diem
};
void nhap(TS *ts, int n)
{for (int i=0;i<n;++i)
   {
      cin.ignore();
      cout<<"\n Thi sinh thu "<<i+1<<endl;
      cout<<"\n Ho ten: ";
      cin.get(ts[i].ht,20);
      cout<<"so bao danh: ";
      cin>>ts[i].sobd;
      cout<<"tong diem: ";
      cin>>ts[i].td;
    }
}
```

```
void sapxep(TS *ts, int n)
{ for (int i=0;i<n-1;++i)
    for (int j=i+1;j<n;++j)
     if  (ts[i].td<ts[j].td)
      {
          TS  tg=ts[i];
              ts[i]=ts[j];
              ts[j]=tg;
      }
}
void xuat(TS *ts, int n)
{   for (int i=0;i<n;++i)
      cout<<ts[i].ht<<setw(6)<<ts[i].td<<endl;
}
```

```
int main(void)
{
   TS *ts;
   int n;
   cout<<"\nNhap so thi sinh n = ";
   cin>>n;
   ts = new TS[n];
   if (ts == NULL)
     {
                    cout << "\n Loi cap phat vung nho";
                    exit(1);
     }
   nhap(ts,n);
   sapxep(ts,n);
   cout<<"\nDanh sach sau khi sap xep: "<<endl;
   xuat(ts,n);
   delete ts;
   return 0;
}
```

Thảo luận:

Viết lại chương trình ở trên, trong đó:

- Hàm nhập phải kiểm tra số báo danh không trùng nhau.
- Viết hàm hoán vị để áp dụng cho hàm sắp xếp.

2.4. Biến tham chiếu (reference variable)

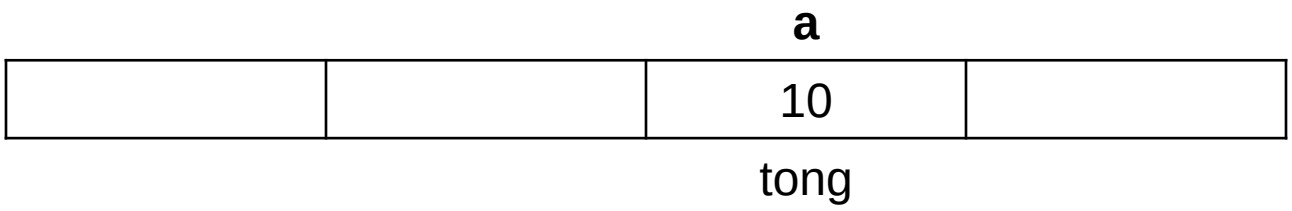
Trong C có 2 loại biến là: Biến giá trị dùng để chứa dữ liệu (nguyên, thực, ký tự,...) và biến con trỏ dùng để chứa địa chỉ. Các biến này đều được cung cấp bộ nhớ và có địa chỉ.

C++ cho phép sử dụng loại biến thứ ba là **biến tham chiếu**. Biến tham chiếu là một tên khác (bí danh) của biến đã định nghĩa trước đó. Cú pháp khai báo biến tham chiếu như sau:

Kiểu &Biến tham chiếu = Biến;

Biến tham chiếu có đặc điểm là nó được dùng làm bí danh cho một biến (kiểu giá trị) nào đó và sử dụng vùng nhớ của biến này. Xem hình sau:

```
int a =10; int &tong = a;  //tong la bi danh của biến a
```



Ví dụ: Với câu lệnh: int a;

int &tong=a;

thì *tong* là bí danh của biến a và biến *tong* dùng chung vùng nhớ của biến a. Lúc này, trong mọi câu lệnh, viết *a* hay viết *tong* đều có ý nghĩa như nhau, vì đều truy nhập đến cùng một vùng nhớ. Mọi sự thay đổi đối với biến *tong* đều ảnh hưởng đối với biến a và ngược lại.

Ví dụ:

```
int a = 5 ; int &tong =a;
cout<< tong;     //5
++tong;          //gia tri tong la 6
++a;             //gia tri a la 7
cout<<“ tong = “<<tong;    //tong = 7
++a; //gia tri a la 8
cout<<“ a = “<<a<<“tong = “<<tong;
//a = 8     tong = 8
```

2.5. Hằng tham chiếu

Cú pháp khai báo hằng tham chiếu như sau:

const Kiểu dữ liệu &Biến = Biến/Hằng;

Ví dụ:

```
int n = 10;
const int &m = n;
const int &p = 123;
```

Hằng tham chiếu có thể tham chiếu đến một biến hoặc một hằng.

Chú ý:

Biến tham chiếu và hằng tham chiếu khác nhau ở chỗ: không cho phép dùng hằng tham chiếu để làm thay đổi giá trị của vùng nhớ mà nó tham chiếu.

Ví dụ:

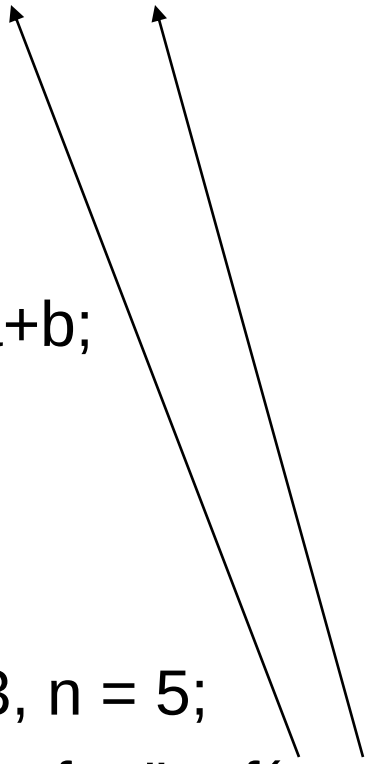
```
int y=12, z;
const int &p = y  //Hằng tham chiếu p tham chiếu đến
                  // biến y
p = p + 1;        //Sai, trình biên dịch sẽ thông báo lỗi
```

2.6. Truyền tham số cho hàm theo tham chiếu

Trong C chỉ có một cách truyền dữ liệu cho hàm là truyền theo theo giá trị. Chương trình sẽ tạo ra các bản sao của các tham số thực sự trong lời gọi hàm và sẽ thao tác trên các bản sao này chứ không xử lý trực tiếp với các tham số thực sự.

Tuy nhiên, nhiều khi ta lại muốn những tham số đó thay đổi khi thực hiện hàm trong chương trình. **C++ cung cấp thêm cách truyền dữ liệu cho hàm theo tham chiếu bằng cách dùng đối là tham chiếu.** Cách làm này có ưu diểm là không cần tạo ra các bản sao của các tham số, do dó tiết kiệm bộ nhớ và thời gian chạy máy. Mặt khác, hàm này sẽ thao tác trực tiếp trên vùng nhớ của các tham số, do đó dễ dàng thay đổi giá trị các tham số khi cần.

```
//…
int f(int &a, int b)
{
   ++a;
   ++b;
   return a+b;
}
int main()
{
  int m = 3, n = 5;
  cout<<"\n f = "<<f(m,n);          //    f=
  cout<<"\n m = "<<m;                //    m=
  cout<<"\n n = "<<n;                //    n =
 return 0;
 }
```



```
// Cho biet ket qua cua ham main
int f(int a, int &b)
{
   ++a;
   ++b;
   return a+b;
}
int main()
{
  int m = 3, n = 5;
  cout<<"\n f = "<<f(m,n);              // f =
  cout<<"\n m = "<<m;                   // m =
  cout<<"\n n = "<<n;                   // n =
 return 0;
 }
```

```
// Cho biet ket qua cua ham main
int f(int &a, int &b)
{
   ++a;
   ++b;
   return a+b;
}
int main()
{
  int m = 3, n = 5;
  cout<<"\n f = "<<f(m,n);              // f =
  cout<<"\n m = "<<m;                   // m =
  cout<<"\n n = "<<n;                   // n =
 return 0;
 }
```

```
// Cho biet ket qua cua ham main
int f(int a, int b)
{
   ++a;
   ++b;
   return a+b;
}
int main()
{
  int m = 3, n = 5;
  cout<<"\n f = "<<f(m,n);              // f =
  cout<<"\n m = "<<m;                   // m =
  cout<<"\n n = "<<n;                   // n =
 return 0;
 }
```

```
void Swap(int &a,int &b)
{ int tam;
  tam = a;
  a = b;
  b = tam;
}
int main()
{ int m = 2, n = 5;
  cout<<"\n Truoc khi hoan vi: ";
  cout<<"m = "<<m<<" , n = "<<n;
  Swap(m,n);
  cout<<"\n Sau khi hoan vi:   ";
  cout<<"m = "<<m<<" , n = "<<n;
  return 0;
}
```

Nếu dung con trỏ:

```
void Swap(int *a,int *b)
{
   int tam;
   tam = *a;
   *a = *b;
   *b = tam;
}
```

2.7. Hàm trả về giá trị tham chiếu

C++ cho phép hàm trả về giá trị là một tham chiếu, lúc này định nghĩa của hàm có dạng như sau :

```
Kiểu     &Tên hàm(...)
  {      //thân hàm
         return   <biến phạm vi toàn cục>;
    }
```

Chú ý:

- Biểu thức được trả lại trong câu lệnh *return* phải là tên của một biến xác định từ bên ngoài hàm, bởi vì khi đó mới có thể sử dụng được giá trị của hàm.
- Nếu ta trả về một tham chiếu đến một biến cục bộ khai báo bên trong hàm, biến cục bộ này sẽ bị mất đi khi kết thúc thực hiện hàm. Do vậy tham chiếu của hàm sẽ không còn ý nghĩa nữa.

- Khi giá trị trả về của hàm là tham chiếu, ta có thể gặp câu lệnh gán hơi khác thường, trong đó vế trái là một lời gọi hàm chứ không phải là tên của một biến. Điều này hoàn toàn hợp lý, bởi vì bản thân hàm đó có giá trị trả về là một tham chiếu. Nói cách khác, vế trái của lệnh gán có thể là lời gọi đến một hàm có giá trị trả về là một tham chiếu. Xem các ví dụ sau:

Ví dụ 2.9.

```
#include <iostream>
using namespace std;
int z;  //z la bien toan cuc
int &f()//ham tra ve mot bi danh cua bien toan cuc z
   {  return z;   }
int  main()
{
    z = 10;
   cout<<"\n f() = "<< f();
   f() = 20;   //luc nay z = ?
   return 0;
}
```

Ví dụ 2.10

```
#include <iostream>
using namespace std;
int &max(int &a, int &b)  //hàm trả về max của a và b
{
  return a>b ? a:b;
}
int main()
{
  int a =7, b = 10, c= 20;
  cout << "Max a,b : "<<max(b,a) << endl; // Max a,b : 10
  max(b,a)++; //tang b lên 1
  cout << " a = "<< a <<"  b = "<<b <<endl;  //a = 7 b = 11
  max(b,c)=5;  //gan lai gia tri 5 cho bien c
  cout << " a = "<< a <<"  b = "<<b <<"  c = "<<c<< endl;//a=7   b=11 c=5
  return 0;
 }
```

Thảo luận: Cho biết kết quả của chương trình trên. Giải thích

Xem chương trình sau:

```
#include <iostream>
using namespace std;
int &max(int &a, int &b)
{
  return a>b ? a:b;
}
int main()
{
  int a =10, b = 4, c= 2;
  cout << "Max a,b : "<<max(b,a) << endl; // Max a,b:
  max(b,a)++; //
  cout << " a = "<< a <<"  b = "<<b <<endl;  //a =   b =
  max(b,c)=5;
  cout << " a = "<< a <<"  b = "<<b <<"  c = "<<c<< endl;//a=  b= c=
  return 0;
 }
```

Thảo luận: Cho biết kết quả của chương trình trên. Giải thích

2.8. Hàm với tham số có giá trị mặc định

C++ cho phép xây dựng hàm với các tham số được khởi gán giá trị mặc định. Quy tắc xây dựng hàm với tham số mặc định như sau:

(i) Các tham số có giá trị mặc định cần là các tham số **cuối cùng** tính từ trái qua phải.

(ii) Nếu chương trình sử dụng khai báo nguyên mẫu hàm thì các tham số mặc định cần được khởi gán trong nguyên mẫu hàm, không được khởi gán lại cho các tham số mặc định trong dòng đầu của định nghĩa hàm.

Ví dụ:

```
void f(int a, float x, char *st=”TRUNG TAM”, int b=1);  //nguyên mẫu hàm f
void f(int a, float x, char *st, int b)
        {
            //Các câu lệnh
        }
```

Chú ý: Đối với các hàm có đối số mặc định thì lời gọi hàm cần viết theo quy định: Các tham số vắng mặt trong lời gọi hàm tương ứng với các đối số mặc định cuối cùng (tính từ trái sang phải), ví dụ với hàm:

void f(int a, float x, char *st=”TRUNG TAM”, int b=1);

thì các lời gọi hàm đúng:

f(3,3.4,”TIN HOC”,10);//Đầy đủ tham số

f(3,3.4,”ABC”); //Thiếu 1 tham số cuối

f(3,3.4); //Thiếu 2 tham số cuối

Các lời gọi hàm sai: f(3);

Các khai báo sau **đúng hay sai**, **giải thích?**

void f1(int a=1, int b, int c); //1 S

void f1(int a=1, int b=2, int c); //2 S

void f1(int a=1, int b, int c=2); //3 S

void f1(int a=1, int b=2, int c=3); //4 Đ

void f1(int a, int b=2, int c=3); //5 Đ

Xem chương trình sau:

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

void ht(char *dc="TRUNG TAM",int n=5)
 {
   for(int i=0;i<n;++i)
      cout<<"\n" <<dc;
 }

int  main()
  {
  ht();//
  ht("ABC",3);        //?
  ht("DEF");          // ?
  return 0;
}
```

Thảo luận: Cho biết kết quả thực thi chương trình trên. Giải thích.

2.10. Hàm tải bội

- Các hàm tải bội (còn gọi là hàm quá tải) là các hàm có cùng một tên và có **tập đối khác nhau** (về **số lượng các đối hoặc kiểu**).
- Khi gặp lời gọi các hàm tải bội thì trình biên dịch sẽ căn cứ vào số lượng và kiểu các tham số để gọi hàm có đúng tên và đúng các đối số tương ứng.

Ví dụ 2.14 Chương trình dùng hàm tải bội để thực hiện việc tính tích vô hướng của 2 vec tơ và tích một số nguyên với một vec tơ.

Ví dụ 2.15 Chương trình tìm **max** của một dãy số nguyên và **max** của một dẫy số thực. Trong chương trình có 6 hàm: hai hàm dùng để nhập dãy số nguyên và dãy số thực có tên chung là nhapds, bốn hàm: tính max 2 số nguyên, tính max 2 số thực, tính max của dãy số nguyên, tính max của dãy số thực được đặt chung một tên là Max.

CÂU HỎI VÀ BÀI TẬP

Xem trang 45-46 Giáo trình Lập trình Hướng đối tượng