

MỘT SỐ TIỆN ÍCH VÀ MỞ RỘNG CỦA C++ SO VỚI C

MỤC TIÊU CỦA BÀI NÀY GIÚP NGƯỜI HỌC

- Nhập/xuất dữ liệu sử dụng toán tử cin và cout
- Viết chú thích trên một dòng, khai báo biến ở mọi nơi, cấp phát và thu hồi bộ nhớ động sử dụng toán new và delete,
- Giải các bài tập có sử dụng kỹ thuật chồng hàm, tham số ngầm định.

A/ TÓM TẮT LÝ THUYẾT

- C++ là một sự mở rộng của C, do đó có thể sử dụng một chương trình biên dịch C++ để dịch và thực hiện các chương trình viết bằng C
- C yêu cầu các chú thích nằm giữa /* và */. C++ cho phép tạo một chú thích bắt đầu bằng “//” cho đến hết dòng
- C++ cho phép khai báo tùy ý. Thậm chí có thể khai báo biến trong phần khởi tạo của câu lệnh lặp for
- C++ cho phép truyền tham số cho hàm bằng tham chiếu. Điều này tương tự như truyền tham biến cho chương trình con trong ngôn ngữ lập trình PASCAL. Trong lời gọi hàm ta dùng tên biến và biến đó sẽ được truyền cho hàm qua tham chiếu. Điều đó cho phép thao tác trực tiếp trên biến được truyền chứ không phải gián tiếp qua biến trở.
- Toán tử new và delete trong C++ được dùng để quản lý bộ nhớ động thay vì các hàm cấp phát động của C
- C++ cho phép người viết chương trình mô tả các giá trị ngầm định cho các tham số của hàm, nhờ đó hàm có thể được gọi với một danh sách các tham số không đủ.
- Toán tử “::” cho phép truy nhập biến toàn cục khi đồng thời sử dụng biến cục bộ và toàn cục cùng tên.
- Có thể định nghĩa các hàm cùng tên với các tham số khác nhau. Hai hàm cùng tên sẽ được phân biệt nhờ giá trị trả về và danh sách kiểu các tham số.

B. MỘT SỐ LƯU Ý (Các lỗi thường gặp, một số thói quen lập trình tốt...)

☛ Các lỗi thường gặp

- Quên đóng */ cho các chú thích
- Khai báo biến sau khi biến được sử dụng
- Sử dụng lệnh return để trả về giá trị nhưng khi định nghĩa hàm lại mô tả hàm kiểu **void** hoặc ngược lại, quên câu lệnh này trong trường hợp hàm yêu cầu giá trị trả về.
- Không có hàm nguyên mẫu cho các hàm
- Bỏ qua khởi tạo cho các biến tham chiếu
- Thay đổi giá trị của các hằng
- Tạo các hàm cùng tên, cùng tham số.

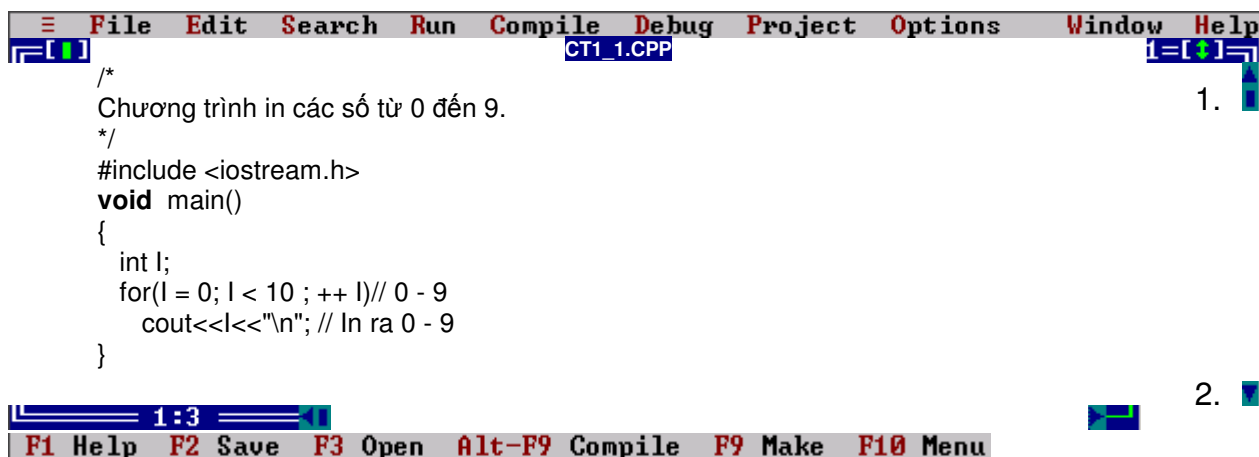
☛ Một số thói quen lập trình tốt

- Sử dụng “//” để tránh lỗi không đóng */ khi chú thích nằm gọn trong một dòng.
- Sử dụng các khả năng vào ra mới của C++ để chương trình dễ đọc hơn.
- Đặt các khai báo biến lên đầu khối lệnh.
- Chỉ dùng từ khóa inline với các hàm “nhỏ”, “không phức tạp”.
- Sử dụng con trỏ để truyền tham số cho hàm khi cần thay đổi giá trị tham số, còn tham chiếu dùng để truyền các tham số có kích thước lớn mà không có nhu cầu thay đổi nội dung.
- Tránh sử dụng biến cùng tên cho nhiều mục đích khác nhau trong chương trình.

C/ BÀI TẬP MẪU

Ví dụ 1: C++ chấp nhận hai kiểu chú thích. Các lập trình viên bằng C đã quen với cách chú thích bằng `/*...*/`. Trình biên dịch sẽ bỏ qua mọi thứ nằm giữa `/*...*/`.

Xét chương trình sau :



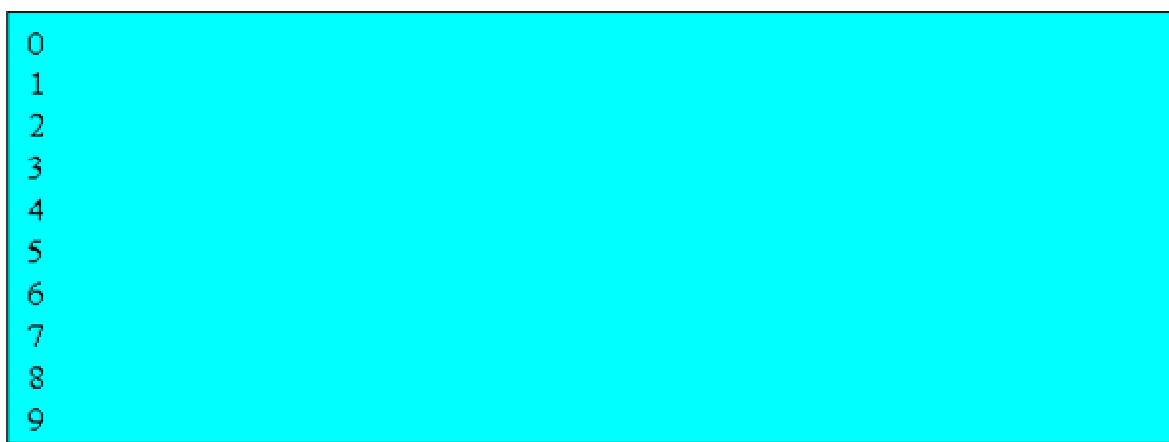
```

/*
Chương trình in các số từ 0 đến 9.
*/
#include <iostream.h>
void main()
{
    int i;
    for(i = 0; i < 10 ; ++ i)// 0 - 9
        cout<<i<<"\n"; // In ra 0 - 9
}

```

Mọi thứ nằm giữa `/*...*/` từ dòng 1 đến dòng 3 đều được chương trình bỏ qua. Chương trình này còn minh họa cách chú thích thứ hai. Đó là cách chú thích bắt đầu bằng `//` ở dòng 8 và dòng 9.

kết quả



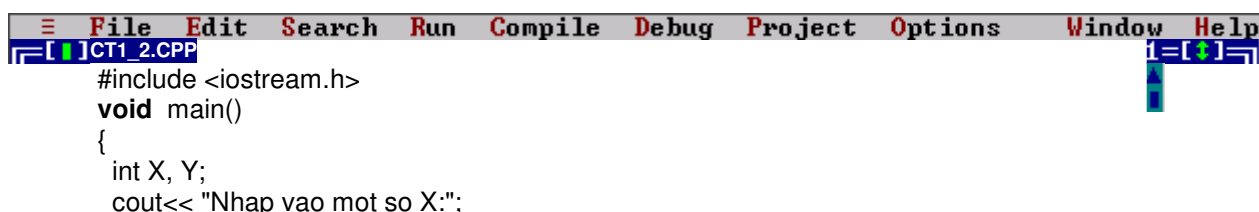
```

0
1
2
3
4
5
6
7
8
9

```

Nói chung, kiểu chú thích `/*...*/` được dùng cho các khối chú thích lớn gồm nhiều dòng, còn kiểu `//` được dùng cho các chú thích một dòng.

Ví dụ 2: Chương trình nhập vào hai số. Tính tổng và hiệu của hai số vừa nhập.



```

#include <iostream.h>
void main()
{
    int X, Y;
    cout<< "Nhập vào một số X:";
}

```

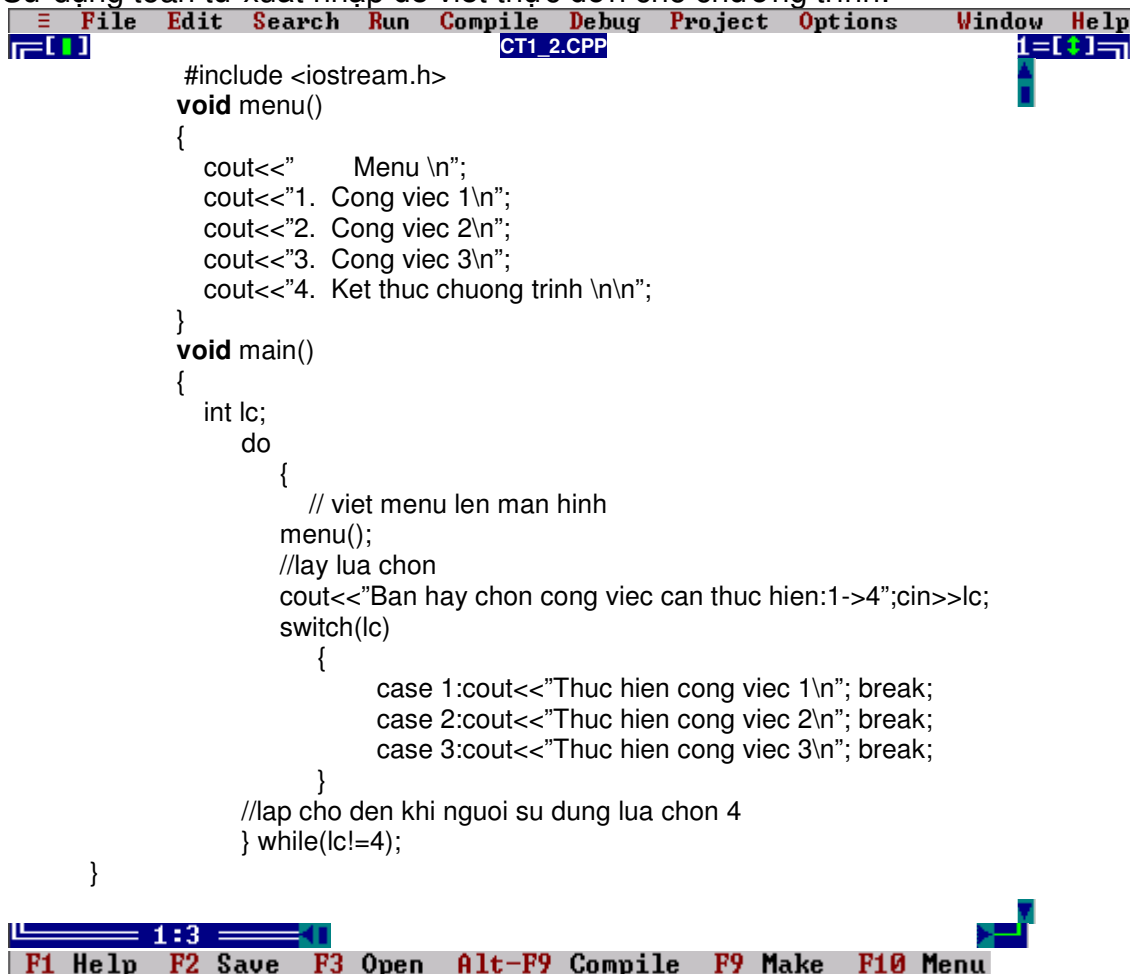
```

cin>>X;
cout<< "Nhap vao mot so Y:";
cin>>Y;
cout<<"Tong cua chung:"<<X+Y<<"\n";
cout<<"Hieu cua chung:"<<X-Y<<"\n";
}

```

**Ví dụ 3:**

Sử dụng toán tử xuất nhập để viết thực đơn cho chương trình:

**Ví dụ 4:**

Tìm lỗi sai của đoạn chương trình sau:

```

int n;
cin>>n;
for(int i=0;i<n;i++)
{ int a[100];
  cin>>a[i];
}
for(i=0;i<n;i++)
  cout<<a[i];

```

Lời giải

Chương trình bị lỗi trong vòng for thứ hai do biến mảng a không được định nghĩa. Mảng a được khai báo trong vòng for thứ nhất chỉ có tầm hoạt động trong vòng for đó mà thôi. Do vậy, chương trình không thể biết ở trong vòng lặp for thứ hai. Chú ý

biến nguyên i được khai báo trong dòng lệnh for có vị trí tương đương với việc khai báo i ở bên ngoài for. Vì vậy, trong vòng for thứ hai ta sử dụng biến i nhưng chương trình không báo lỗi.

Ví dụ 5:

Tìm lỗi sai cho các khai báo prototype hàm dưới đây (các hàm này được khai báo trong cùng một chương trình)

```
int func1(int);           // (1)
float func1(int);         // (2)
int func1(float);         // (3)
void func1(int=0,int);    // (4)
void func2(int,int=0);    // (5)
void func2(int);          // (6)
void func2(float);        // (7)
```

Lời giải:

Trong định nghĩa chồng hàm, trình biên dịch phân biệt các hàm bởi kiểu dữ liệu trả ra của hàm mà chỉ phân biệt bởi danh sách tham số của hàm. Do vậy hàm 1 và hàm 2 bị định nghĩa chồng lên nhau và trình biên dịch báo lỗi. Giữa hàm 2 và hàm 3 không có lỗi bởi chúng khác nhau bởi kiểu dữ liệu của tham số. Trong hàm 4 ta đã sử dụng sai cách truyền giá trị mặc định cho tham số. Không báo giờ truyền giá trị mặc định cho một tham số trước một tham số không được truyền giá trị ngầm định.

Trong cách định nghĩa hai hàm 5 và 6 có sự nhập nhằng. Khi ta gọi hàm func2 với tham số là một số nguyên thì trình biên dịch không biết là sẽ gọi hàm 5 hay hàm 6 bởi vì cả hai hàm này đều được. Trong trường hợp này trình biên dịch cũng thông báo lỗi.

Ví dụ 6:

Tìm lỗi sai (lỗi cú pháp và bộ nhớ) cho chương trình sau:

```
int &ref1()
{
    int a=5;
    return a;
}
int &ref2(int a)
{
    a++;
    return a;
}
int &ref3(int &a)
{
    a++;
    return a;
}
int a=5;
int &r1;
int &r2=22;
int &r3=a;
int &r4=ref3(5);
int &r5=ref3(a);
```

Trả lời:

Trong các hàm có kết quả trả về là một tham chiếu, chúng ta luôn phải chú ý rằng biến được trả lại có giá trị là tham chiếu không bị xóa khỏi bộ nhớ chương trình khi kết thúc thực hiện hàm. Do vậy hai hàm ref1 và ref2 là sai bởi vì nó trả về tham chiếu tới biến mà a lại là biến cục bộ trong ref1 và là tham số trong ref2 chỉ được tạo ra tạm thời

trên stack khi gọi hàm và xoá khỏi stack khi kết thúc hàm. Hàm ref3 không có lỗi vì a là một tham chiếu tới một biến không nằm trong hàm.

Trong khai báo các tham chiếu phải được gắn với một biến nào đó trong bộ nhớ. Do vậy các khai báo r1, r2 là sai. Lời gọi ref3(5) cũng là sai bởi vì tham số cho hàm phải là tham chiếu đến một biến, trong khi đó ta lại truyền vào hằng số.

Ví dụ 7:

Cho biết kết quả thực hiện chương trình sau:

```
#include <iostream.h>
int & foo(int &a,int b)
{
    b+=a;
    if (b>5) a++;
    return a;
}
void main()
{
    int i=2,j=4;
    int k=foo(i,j);
    k++;
    cout<<i<<" "<<j<<" "<<k<<endl;
    int &l=foo(i,j);
    l++ ;
    cout<<i<<" "<<j<<" "<<l<<endl;
}
```

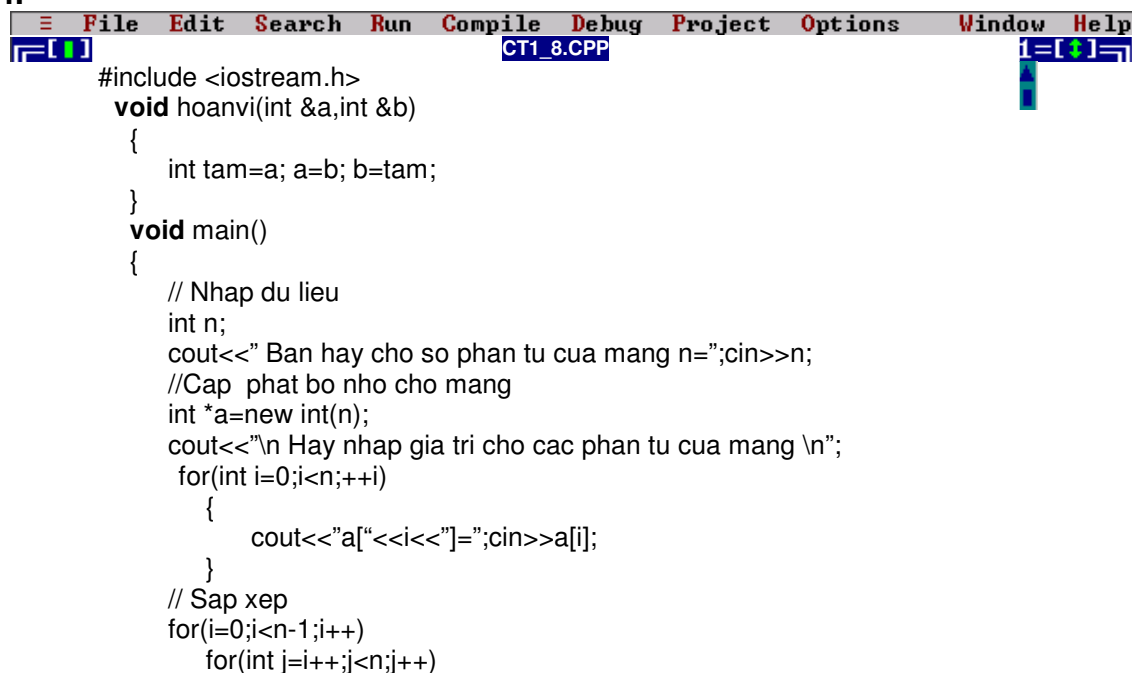
Lời giải:

Trong chương trình trên cần chú ý hai điểm. Thứ nhất là ta truyền vào cho hàm tham chiếu của biến i chứ không phải biến i. Do vậy, mọi thay đổi của tham số này trong hàm là thay đổi tới biến i được tham chiếu tới. Tương tự như vậy với tham chiếu l. Tham chiếu l được xác lập bằng tham chiếu trả ra của hàm chính là tham chiếu tới biến i. Do vậy mọi thay đổi l chính là thay đổi i.

Ví dụ 8:

Viết một hàm hoanvi dùng để hoán vị hai số nguyên. Sau đó viết chương trình nhập và sắp xếp một mảng số nguyên.

Trả lời:



```

File Edit Search Run Compile Debug Project Options Window Help
CT1_8.CPP
#include <iostream.h>
void hoanvi(int &a,int &b)
{
    int tam=a; a=b; b=tam;
}
void main()
{
    // Nhập du lieu
    int n;
    cout<<" Ban hay cho so phan tu cua mang n=";<<cin>>n;
    //Cap phat bo nho cho mang
    int *a=new int(n);
    cout<<"\n Hay nhap gia tri cho cac phan tu cua mang \n";
    for(int i=0;i<n;++i)
    {
        cout<<"a["<<i<<"]=";<<cin>>a[i];
    }
    // Sap xep
    for(i=0;i<n-1;i++)
        for(int j=i++;j<n;j++)

```

```

        if (a[i]>a[j]) hoanvi(a[i],a[j]);
    // In ket qua
    cout<<"\n Cac phan tu cua mang sau khi da sap xep la \n";
    for(i=0;i<n;i++)
        cout<<a[i]<<" ";
    delete a;
}

```



Ví dụ 9: Chương trình tạo một mảng động, khởi động mảng này với các giá trị ngẫu nhiên và sắp xếp chúng.



kết quả

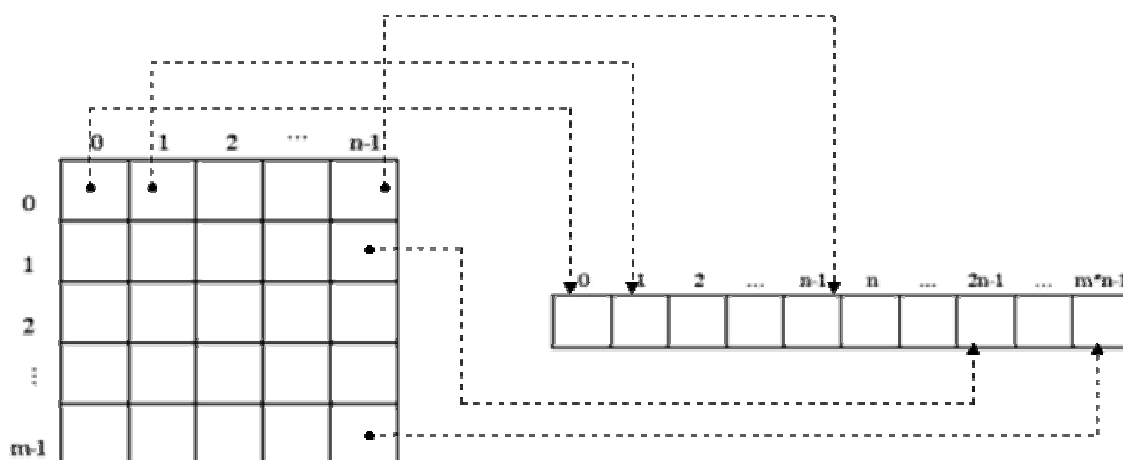
```

Nhap vao so phan tu cua mang: 10
Mang truoc khi sap xep
81 83 11 71 58 7 5 59 74 24
Mang sau khi sap xep
5 7 11 24 58 59 71 74 81 83

```

Ví dụ 10: Chương trình cộng hai ma trận trong đó mỗi ma trận được cấp phát động.

Chúng ta có thể xem mảng hai chiều như mảng một chiều như hình 1.2 dưới đây



Hình 1.2: Mảng hai chiều có thể xem như mảng một chiều.

Gọi X là mảng hai chiều có kích thước m dòng và n cột.

A là mảng một chiều tương ứng.

Nếu $X[i][j]$ chính là $A[k]$ thì $k = i * n + j$

Chúng ta có chương trình như sau :

```

File Edit Search Run Compile Debug Project Options Window Help
CT1_10.CPP
#include <iostream.h>
#include <conio.h>
//prototype
void AddMatrix(int * A,int *B,int *C,int M,int N);
int AllocMatrix(int **A,int M,int N);
void FreeMatrix(int *A);
void InputMatrix(int *A,int M,int N,char Symbol);
void DisplayMatrix(int *A,int M,int N);
int main()
{
    int M,N;
    int *A = NULL,*B = NULL,*C = NULL;
    clrscr();
    cout<<"Nhập số dòng của ma trận:";
    cin>>M;
    cout<<"Nhập số cột của ma trận:";
    cin>>N;
    //Cấp phát vùng nhớ cho ma trận A
    if (!AllocMatrix(&A,M,N))
    { //endl: Xuất ra kí tự xuống dòng ('\n')
        cout<<"Không còn đủ bộ nhớ!"<<endl;
        return 1;
    }
    //Cấp phát vùng nhớ cho ma trận B
    if (!AllocMatrix(&B,M,N))
    {

```

```
        cout<<"Khong con du bo nho!"<<endl;
        FreeMatrix(A);//Giải phóng vùng nhớ A
        return 1;
    }
    //Cấp phát vùng nhớ cho ma trận C
    if (!AllocMatrix(&C,M,N))
    {
        cout<<"Khong con du bo nho!"<<endl;
        FreeMatrix(A);//Giải phóng vùng nhớ A
        FreeMatrix(B);//Giải phóng vùng nhớ B
        return 1;
    }
    cout<<"Nhap ma tran thu 1"<<endl;
    InputMatrix(A,M,N,'A');
    cout<<"Nhap ma tran thu 2"<<endl;
    InputMatrix(B,M,N,'B');
    clrscr();
    cout<<"Ma tran thu 1"<<endl;
    DisplayMatrix(A,M,N);
    cout<<"Ma tran thu 2"<<endl;
    DisplayMatrix(B,M,N);
    AddMatrix(A,B,C,M,N);
    cout<<"Tong hai ma tran"<<endl;
    DisplayMatrix(C,M,N);
    FreeMatrix(A);//Giải phóng vùng nhớ A
    FreeMatrix(B);//Giải phóng vùng nhớ B
    FreeMatrix(C);//Giải phóng vùng nhớ C
    return 0;
}
//Cộng hai ma trận
void AddMatrix(int *A,int *B,int *C,int M,int N)
{
    for(int l=0;l<M*N;++l)
        C[l] = A[l] + B[l];
}
//Cấp phát vùng nhớ cho ma trận
int AllocMatrix(int **A,int M,int N)
{
    *A = new int [M*N];
    if (*A == NULL)
        return 0;
    return 1;
}
//Giải phóng vùng nhớ
void FreeMatrix(int *A)
{
    if (A!=NULL)
        delete [] A;
}
//Nhập các giá trị của ma trận
void InputMatrix(int *A,int M,int N,char Symbol)
{
    for(int l=0;l<M;++l)
        for(int J=0;J<N;++J)
        {
            cout<<Symbol<<"["<<l<<"["<<J<<"]=";
            cin>>A[l*N+J];
        }
}
//Hiển thị ma trận
void DisplayMatrix(int *A,int M,int N)
{
```



```

for(int l=0;l<M;++l)
{
    for(int J=0;J<N;++J)
    {
        out.width(7);//Hien thi canh le phai voi chieu dai 7 ky tu
        cout<<A[l*N+J];
    }
    cout<<endl;
}
}

```

1:3
F1 Help **F2** Save **F3** Open **Alt-F9** Compile **F9** Make **F10** Menu

kết

quả

Nhap so dong cua ma tran: 2	Ma tran thu 1
Nhap so cot cua ma tran: 3	1 2 3
Nhap ma tran thu 1	4 5 6
A[0][0]=1	Ma tran thu 2
A[0][1]=2	7 8 9
A[0][2]=3	10 11 12
A[1][0]=4	Tong hai ma tran
A[1][1]=5	8 10 12
A[1][2]=6	14 16 18
Nhap ma tran thu 2	
B[0][0]=7	
B[0][1]=8	
B[0][2]=9	
B[1][0]=10	
B[1][1]=11	
B[1][2]=12	

D/ BÀI TẬP TỰ GIẢI

Câu hỏi trắc nghiệm

Câu 1: Cho biết giá trị của k sau khi thực hiện đoạn chương trình

```

int i=5,k;
{
    int i=6;
    ::i--;
    k=i;
}
k-=i;

```

Với các kết quả:

- a) k=0 b) k=1 c) k=2 d) k=3

Câu 2: Tìm lời gọi hàm sai cho hàm sau:

```

void func(int i=0,int j=0);
a)func()
b)dunc(1);

```

c)func(1.5,2.5);

d)func(1,2);

Câu 3: Cho biết giá trị của y sau khi thực hiện:

```
int &foo(int &a)
```

```
{    a++;
```

```
    return a;
```

```
}
```

```
int i=5;
```

```
int &r=foo(i);
```

```
r++;
```

a) i=5;

b) i=6;

c) i=7;

d) không câu nào đúng

Câu 4: Tìm giá trị của x, y:

```
void test(int &a, int b)
```

```
{    a+=b;
```

```
    b=a;
```

```
}
```

```
int x=1,y=2;
```

```
test(x,y);
```

a) x=1,y=2;

b) x=1,y=3

c) x=3,y=2

d) x=3,y=3

Bài tập tự giải

Bài 1.1: Anh (chị) hãy viết lại chương trình sau bằng cách sử dụng lại các dòng nhập/xuất trong C++.

```
/*
```

Chương trình tìm mẫu chung nhỏ nhất

```
*/
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int a,b,i,min;
```

```
    printf("Nhap vao hai so:");
```

```
    scanf("%d%d",&a,&b);
```

```
    min=a>b?b:a;
```

```

for(i = 2; i < min; ++i)

    if (((a%i)==0)&&((b%i)==0)) break;

    if(i==min)

    {

        printf("Khong co mau chung nho nhat");

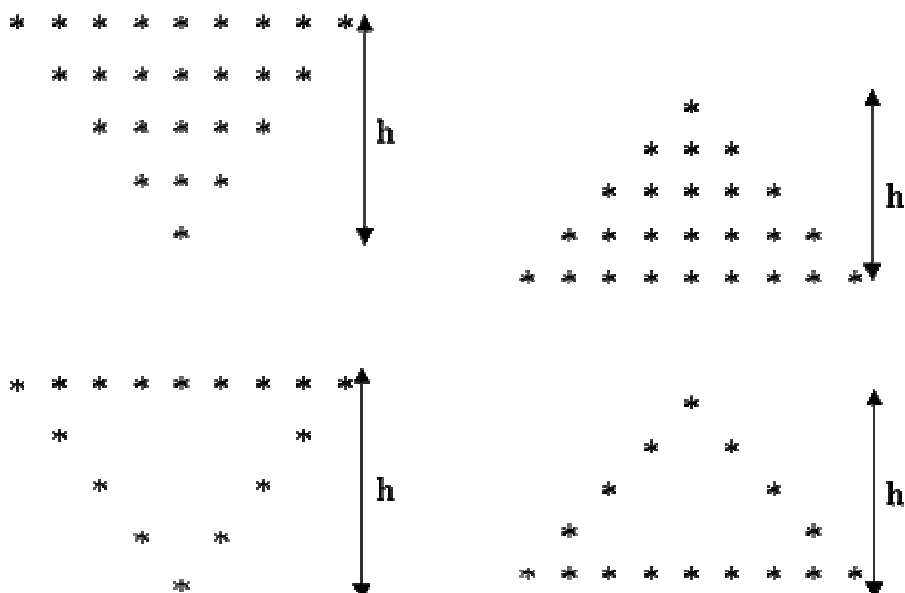
    }

    printf("Mau chung nho nhat la %d\n", i);

}

```

Bài 1.2: Viết chương trình nhập vào số nguyên dương h ($2 < h < 23$), sau đó in ra các tam giác có chiều cao là h như các hình sau:



Bài 1.3: Một tam giác vuông có thể có tất cả các cạnh là các số nguyên. Tập của ba số nguyên của các cạnh của một tam giác vuông được gọi là bộ ba Pitago. Đó là tổng bình phương của hai cạnh bằng bình phương của cạnh huyền, chẳng hạn bộ ba Pitago (3, 4, 5). Viết chương trình tìm tất cả các bộ ba Pitago như thế sao cho tất cả các cạnh không quá 500.

Bài 1.4: Viết chương trình in bảng của các số từ 1 đến 256 dưới dạng nhị phân, bát phân và thập lục phân tương ứng.

Bài 1.5: Viết chương trình nhập vào một số nguyên dương n . Kiểm tra xem số nguyên n có thuộc dãy Fibonacci không?

Bài 1.6: Viết chương trình nhân hai ma trận $A_{m \times n}$ và $B_{n \times p}$. Mỗi ma trận được cấp phát động và các giá trị của chúng phát sinh ngẫu nhiên (Với m , n và p nhập từ bàn phím).

Bài 1.7: Viết chương trình tạo một mảng một chiều động có kích thước là n (n nhập từ bàn phím). Các giá trị của mảng này được phát sinh ngẫu nhiên trên đoạn [a, b] với a và b đều nhập từ bàn phím. Hãy tìm số dương nhỏ nhất và số âm lớn nhất trong mảng; nếu không có số dương nhỏ nhất hoặc số âm lớn nhất thì xuất thông báo "không có số dương nhỏ nhất" hoặc "không có số âm lớn nhất".

Bài 1.8: Anh (chị) hãy viết một hàm tính bình phương của một số. Hàm sẽ trả về giá trị bình phương của tham số và có kiểu cùng kiểu với tham số.

Bài 1.9: Trong ngôn ngữ C, chúng ta có hàm chuyển đổi một chuỗi sang số, tùy thuộc vào dạng của chuỗi chúng ta có các hàm chuyển đổi sau :

■ `int atoi(const char *s);`

Chuyển đổi một chuỗi s thành số nguyên kiểu int.

■ `long atol(const char *s);`

Chuyển đổi một chuỗi s thành số nguyên kiểu long.

■ `double atof(const char *s);`

Chuyển đổi một chuỗi s thành số thực kiểu double.

Anh (chị) hãy viết một hàm có tên là `atou` (ascii to number) để chuyển đổi chuỗi sang các dạng số tương ứng.

Bài 1.10: Anh chị hãy viết các hàm sau:

■ Hàm `ComputeCircle()` để tính diện tích s và chu vi c của một đường tròn bán kính r. Hàm này có prototype như sau:

`void ComputeCircle(float & s, float &c, float r = 1.0);`

■ Hàm `ComputeRectangle()` để tính diện tích s và chu vi p của một hình chữ nhật có chiều cao h và chiều rộng w. Hàm này có prototype như sau:

`void ComputeRectangle(float & s, float &p, float h = 1.0, float w = 1.0);`

■ Hàm `ComputeTriangle()` để tính diện tích s và chu vi p của một tam giác có ba cạnh a,b và c. Hàm này có prototype như sau:

`void ComputeTriangle(float & s, float &p, float a = 1.0, float b = 1.0, float c = 1.0);`

■ Hàm `ComputeSphere()` để tính thể tích v và diện tích bề mặt s của một hình cầu có bán kính r. Hàm này có prototype như sau:

`void ComputeSphere(float & v, float &s, float r = 1.0);`

■ Hàm `ComputeCylinder()` để tính thể tích v và diện tích bề mặt s của một hình trụ có bán kính r và chiều cao h . Hàm này có prototype như sau:

```
void ComputeCylinder(float & v, float &s, float r = 1.0 , float h = 1.0);
```

Bài 1.11: Viết chương trình quản lý điểm học sinh với cấu trúc danh sách nối đơn. Trong chương trình sử dụng toán tử vào ra và toán tử `new` để cấp phát bộ nhớ động.

Bài 1.12: Viết một hàm thực hiện việc sắp xếp một mảng số nguyên theo chiều tăng dần hoặc giảm dần. Hàm này tự động mặc định kiểu sắp xếp theo chiều tăng dần.

Bài 1.13: Viết một hàm giải phương trình bậc hai. Hàm này trả lại thông báo rằng phương trình có nghiệm hay không có nghiệm kép. Nếu có nghiệm thì nghiệm sẽ được lưu vào tham số $x1$, $x2$ và được truyền như là tham biến.

Bài 1.14:Viết một hàm tìm vị trí xuất hiện đầu tiên của một từ khoá trong một chuỗi. Hàm này trả lại vị trí tìm thấy của từ khoá trong chuỗi(bắt đầu từ 0) và thay đổi con trỏ chuỗi được truyền vào thành vị trí của ký tự ngay sau ký tự cuối cùng của từ khoá. Từ khoá cần tìm được đưa vào như là một tham số và có một giá trị mặc định.

ĐỐI TƯỢNG VÀ LỚP (CLASS AND OBJECT)

MỤC TIÊU CỦA BÀI NÀY GIÚP NGƯỜI HỌC

- Phân tích được khái niệm đóng gói dữ liệu
- Khai báo và sử dụng một lớp
- Khai báo và sử dụng đối tượng.
- Sử dụng hàm thiết lập và hàm huỷ bỏ
- Khai báo và sử dụng hàm thiết lập sao chép
- Vai trò của hàm thiết lập ngầm định

A/ NHẮC LẠI LÝ THUYẾT

Trong C++, tên cấu trúc là một kiểu dữ liệu không cần kèm theo từ khoá **struct**.

Lớp cho phép người lập trình mô tả các đối tượng thực tế với các thuộc tính và hành vi. Trong C++ thường sử dụng từ khoá **class** để khai báo một lớp. Tên lớp là một kiểu dữ liệu dùng khi khai báo các đối tượng thuộc lớp (*các thể hiện cụ thể của lớp*).

Thuộc tính của đối tượng trong một lớp được mô tả dưới dạng các biến thể hiện. Các hành vi là các hàm thành phần bên trong lớp.

Có hai cách định nghĩa các hàm thành phần của một lớp; khi định nghĩa hàm thành phần bên ngoài khai báo lớp phải đặt trước tên hàm thành phần tên của lớp và toán tử "::" để phân biệt với các hàm tự do cùng tên. Chỉ nên định nghĩa hàm thành phần bên trong khai báo lớp khi nó không quá phức tạp để cho chương trình dễ đọc.

Có thể khai báo và sử dụng các con trỏ đối tượng, tham chiếu đối tượng.

Hai từ khoá **public** và **private** dùng để chỉ định thuộc tính truy nhập cho các thành phần (dữ liệu/hàm) khai báo bên trong lớp.

Thành phần bên trong lớp được khai báo **public** có thể truy nhập từ mọi hàm khai báo một đối tượng thuộc lớp đó.

Thành phần **private** trong một đối tượng chỉ có thể truy nhập được bởi các hàm thành phần của đối tượng hoặc các hàm thành phần của lớp dùng để tạo đối tượng (ở đây tính cả trường hợp đối tượng là tham số của hàm thành phần)

Hai hàm thành phần đặc biệt của một lớp gọi là hàm thiết lập và hàm huỷ bỏ. Hàm thiết lập được gọi tự động (ngầm định) mỗi khi một đối tượng được tạo ra và hàm huỷ bỏ được gọi tự động khi đối tượng hết thời gian sử dụng.

Hàm thiết lập có thuộc tính **public**, cùng tên với tên lớp nhưng không có giá trị trả về.

Một lớp có ít nhất hai hàm thiết lập: hàm thiết lập sao chép ngầm định và hàm thiết lập do người lập trình thiết lập (nếu không mô tả tường minh thì đó là hàm thiết lập ngầm định).

Hàm huỷ bỏ cũng có thuộc tính **public**, không tham số, không giá trị trả về và có tên bắt đầu bởi ~ theo sau là tên của lớp.

Bên trong phạm vi lớp (định nghĩa của các hàm thành phần), các thành phần của lớp được gọi theo tên. Trường hợp có một đối tượng toàn cục cùng tên, muốn xác định đối tượng ấy phải sử dụng toán tử "::".

Lớp có thể chứa các thành phần dữ liệu là các đối tượng của lớp khác. Các đối tượng này phải được khởi tạo trước đối tượng tương ứng của lớp bao.

Mỗi đối tượng có một con trỏ chỉ đến bản thân nó, ta gọi đó là con trỏ this. Con trỏ này có thể được sử dụng tường minh hoặc ngầm định để tham xác định các thành phần bên trong đối tượng. Thông thường người ta sử dụng this dưới dạng ngầm định.

Hàm bạn của một lớp là hàm không thuộc lớp nhưng có quyền truy nhập tới các thành phần **private** của lớp.

Khai báo bạn bè có thể khai báo bất kỳ chỗ nào trong khai báo lớp.

B. MỘT SỐ LƯU Ý (Các lỗi thường gặp, một số thói quen lập trình tốt...)

Các lỗi thường gặp

- Quên dấu “;” ở cuối khai báo lớp
- Khởi tạo các thành phần giá trị trong khai báo lớp
- Định nghĩa chồng một hàm thành phần bằng một hàm không thuộc lớp
- Truy nhập đến các thành phần riêng của lớp từ bên ngoài phạm vi lớp
- Khai báo giá trị trả về cho hàm thiết lập và hàm huỷ bỏ
- Khai báo hàm huỷ bỏ có tham số, định nghĩa chồng hàm huỷ bỏ
- Gọi tương minh hàm thiết lập và hàm huỷ bỏ
- Gọi các tham thành phần bên trong hàm thiết lập

Một số thói quen lập trình tốt

- Nhóm tất cả các thành phần có cùng thuộc tính truy nhập ở một nơi trong khai báo lớp, nhờ vậy mỗi từ khoá mô tả truy nhập chỉ được xác định một lần. Khai báo lớp vì vậy dễ đọc hơn. Theo kinh nghiệm, để các thành phần **private** trước tiên rồi đến các thành phần **protected**, cuối cùng là từ khoá **public**.
- Định nghĩa tất cả các hàm thành phần bên ngoài khai báo lớp. Điều này nhằm phân biệt giữa hai phần giao diện và phần cài đặt lớp.
- Sử dụng các tiền xử lý **#ifndef**, **#define**, **#endif** để cho các tập tin tiêu đề chỉ xuất hiện một lần bên trong chương trình nguồn.
- Phải định nghĩa các hàm thiết lập để đảm bảo rằng các đối tượng đều được khởi tạo nội dung một cách đúng đắn.

C/ BÀI TẬP MẪU

Ví dụ 1: Định nghĩa một lớp mô tả và xử lý các điểm trên màn hình đồ hoạ. Với tên lớp là **point**

Lời giải

- + Các thuộc tính của lớp
 - int x; // hoành độ (cột)
 - int y; // tung độ (hàng)
 - int m; // màu
- + Các phương thức
 - Nhập dữ liệu một điểm
 - Hiện thị một điểm
 - Ẩn một điểm

Lớp điểm được xây dựng như sau:

```
class point
{
    private:
        int x,y,m;
    public:
        void nhapsl();
        void hien();
        void an()
        {
            putpixel(x,y,getbkcolor());
        }
        void point::nhap()
        {
            cout<<"\n Nhập hoành độ (cột) và tung độ (hàng) củ điểm:"; cin>>x>>y;
            cout<<" Nhập màu của điểm:";cin>>m;
        }
        void point::hien()
```

```

{
    int mau_ht;
    mau_ht=getcolor();
    putpixel(x,y,m);
    setcolor(mau_ht);
}

```

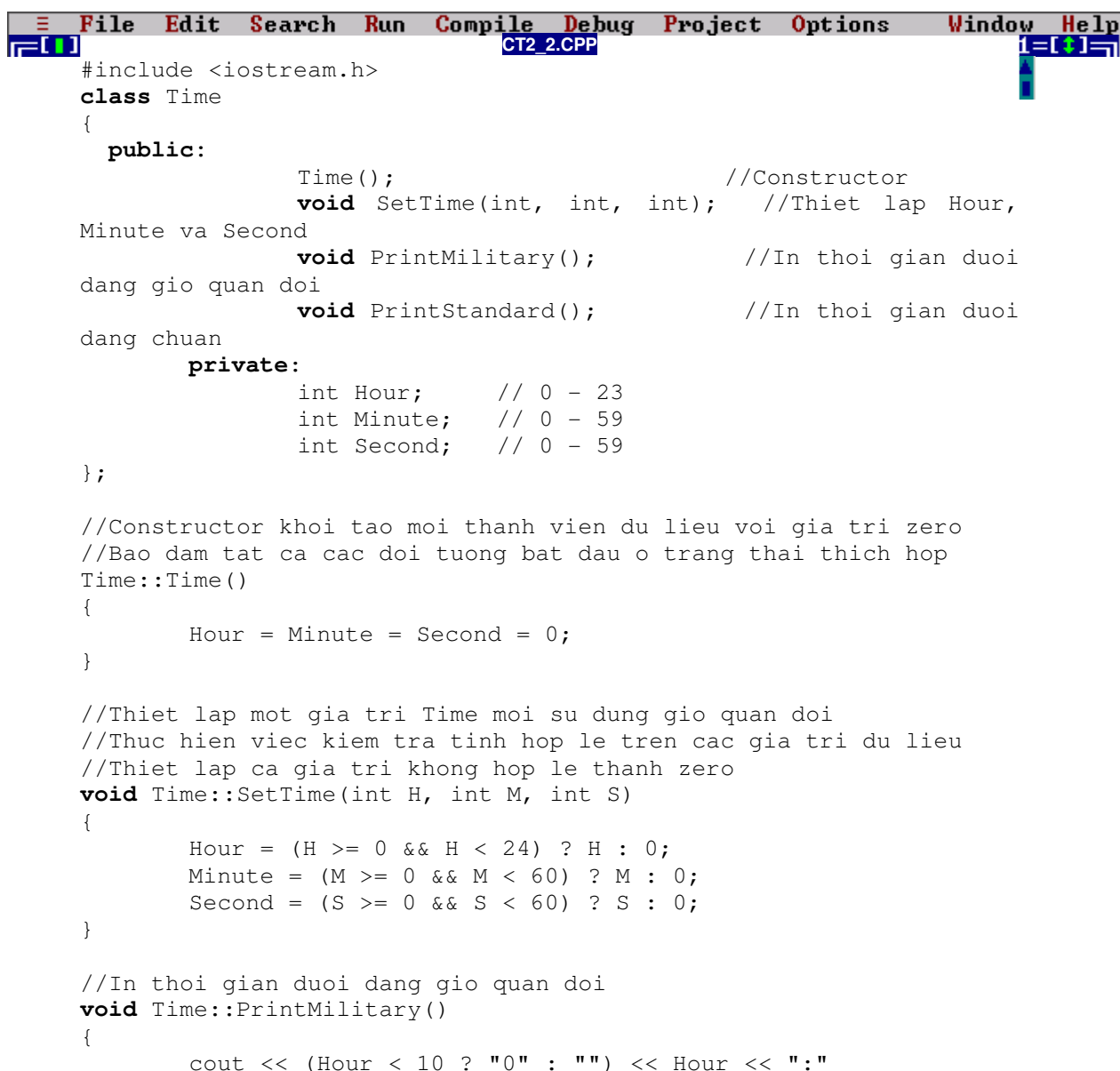
Nhận xét:

+ Trong cả ba phương thức(dù viết trong hay viết ngoài định nghĩa lớp) đều được truy nhập đến các thuộc tính x,y và m của lớp.

+ Các phương thức viết bên trong định nghĩa lớp (như phương thức an()) được viết như một hàm bình thường.

+Khi xây dựng các phương thức bên ngoài lớp, cần dùng thêm tên lớp và toán tử phạm vi :: đặt ngay trước tên phương thức để quy định rõ đây là phương thức của lớp nào.

Ví dụ 2: Chúng ta xây dựng kiểu cấu trúc *Time* với ba thành viên số nguyên: *Hour*, *Minute* và *second*. Chương trình định nghĩa một cấu trúc *Time* gọi là *DinnerTime*. Chương trình in thời gian dưới dạng giờ quân đội và dạng chuẩn.



```

File Edit Search Run Compile Debug Project Options Window Help
CT2_2.CPP
#include <iostream.h>
class Time
{
public:
    Time(); //Constructor
    void SetTime(int, int, int); //Thiet lap Hour,
Minute va Second
    void PrintMilitary(); //In thoi gian duoi
dang gio quan doi
    void PrintStandard(); //In thoi gian duoi
dang chuan
private:
    int Hour; // 0 - 23
    int Minute; // 0 - 59
    int Second; // 0 - 59
};

//Constructor khoi tao moi thanh vien du lieu voi gia tri zero
//Bao dam tat ca cac doi tuong bat dau o trang thai thich hop
Time::Time()
{
    Hour = Minute = Second = 0;
}

//Thiet lap mot gia tri Time moi su dung gio quan doi
//Thuc hien vieckiem tra tinh hop le tren cac gia tri du lieu
//Thiet lap ca gia tri khong hop le thanh zero
void Time::SetTime(int H, int M, int S)
{
    Hour = (H >= 0 && H < 24) ? H : 0;
    Minute = (M >= 0 && M < 60) ? M : 0;
    Second = (S >= 0 && S < 60) ? S : 0;
}

//In thoi gian duoi dang gio quan doi
void Time::PrintMilitary()
{
    cout << (Hour < 10 ? "0" : "") << Hour << ":"

```



```

        << (Minute < 10 ? "0" : "") << Minute << ":"
        << (Second < 10 ? "0" : "") << Second;
    }

    //In thời gian dưới dạng chuẩn
    void Time::PrintStandard()
    {
        cout << ((Hour == 0 || Hour == 12) ? 12 : Hour % 12)
            << ":" << (Minute < 10 ? "0" : "") << Minute
            << ":" << (Second < 10 ? "0" : "") << Second
            << (Hour < 12 ? " AM" : " PM");
    }

    int main()
    {
        Time T;

        cout << "The initial military time is ";
        T.PrintMilitary();
        cout << endl << "The initial standard time is ";
        T.PrintStandard();

        T.SetTime(13, 27, 6);
        cout << endl << endl << "Military time after SetTime is ";
        T.PrintMilitary();
        cout << endl << "Standard time after SetTime is ";
        T.PrintStandard();

        T.SetTime(99, 99, 99); //Thu thiết lập giá trị không hợp
le
        cout << endl << endl << "After attempting invalid
settings:"
            << endl << "Military time: ";
        T.PrintMilitary();
        cout << endl << "Standard time: ";
        T.PrintStandard();
        cout << endl;
        return 0;
    }

```



kết quả

Dinner will be held at 18:30:00 military time,
which is 6:30:00 PM standard time.

Time with invalid values: 29:73:103

Ví dụ 3

Nhập một ngày tháng năm từ bàn phím sau đó in ra màn hình.

Lời giải


```
#include <iostream.h>
#include <conio.h>
#define FALSE 0
#define TRUE !FALSE
char* Thang[]={ "", "gieng", "hai", "ba", "bon", "nam", "sau", "bay", "tam",
                "chin", "muoi", "muoi mot", "chap" };
int NgayThang[]={0,31,28,31,30,31,30,31,31,30,31,30,31};
class CDate
{
private:
    int mNgay,mThang,mNam;
    int laNamNhuan(int);
public:
    void nhap();
    int hopLe();
    void in();
};
void CDate::nhap()
{
    cout<<endl<<"Ngay: ";cin>>mNgay;
    cout<<endl<<"Thang: "; cin>>mThang;
    cout<<endl<<"Nam: ";cin>>mNam;
}
int CDate::hopLe()
{
    if ((mThang<1)|| (mThang>12))
        return FALSE;
    else
    {
        if ((mNgay>=1)&&(mNgay<=NgayThang[mThang]))
            return TRUE;
        else if ((mNgay==29)&&laNamNhuan(mNgay))
            return TRUE;
        else
            return FALSE;
    }
}
int CDate::laNamNhuan(int nam)
{
    if (((nam%400)==0)||(((nam%4)==0)&&((nam%100)!=0)))
        return TRUE;
    else
        return FALSE;
}
void CDate::in()
{
    cout<<endl<<"Ban da nhap vao ngay "<<mNgay;
    cout<<" thang "<<Thang[mThang];
    cout<<" nam "<<mNam;
}
void main()
{
    CDate ngay;
    ngay.nhap();
    if (ngay.hopLe())
        ngay.in();
    else
```

```
        cout<<"BAN NHAP NGAY KHONG HOP LE";  
        getch();  
    }
```



Ví dụ 4

Chỉ ra các cách khai báo đối tượng có thể cho các lớp đối tượng dưới đây: **class A**

```
{  
};  
class B  
{  
    B(int, int);  
    public:  
    B(int=0);  
};  
class C  
{  
    C(C&);  
    public:  
    C();  
};  
class D  
{  
    public:  
    D(D&);  
};
```

Lời giải

Về lớp A

Cách 1: Sử dụng hàm thiết lập ngầm định

A a; hoặc A a();

Cách 2: Sử dụng hàm thiết lập sao chép mặc định. Giả sử a là một đối tượng của lớp A đã được khai báo trước. Ta có thể khai báo đối tượng a1 như sau:

A a1(a); hoặc A a1=a;

Nhận xét:

Khi trong lớp không có một khai báo hàm thiết lập nào thì trình biên dịch tự động tạo ra một hàm thiết lập mặc định cho lớp đó. Do vậy ta có thể sử dụng khai báo đối tượng theo cách 1 cho lớp A. Hai cách viết khai báo

A a1(a); và A a1=a; là hoàn toàn giống nhau, chúng đều sử dụng hàm thiết lập sao chép để khởi tạo đối tượng.

Về lớp B

Cách 1: Sử dụng hàm thiết lập B(int). Ví dụ:

B b(5);

Cách 2: Sử dụng hàm thiết lập B(int) với tham số ngầm định là 0.

B b; tương đương với B b(0);

Cách 3: Sử dụng hàm thiết lập sao chép tương tự như lớp A.

B b1=b;

Nhận xét

Chúng ta chỉ có thể khai báo đối tượng theo các hàm thiết lập có thuộc tính quyền truy nhập là **public**

Trong hàm thiết lập cũng có thể sử dụng tham số ngầm định giống như các hàm thành phần khác.

Về lớp C:

Chỉ có thể khai báo đối tượng theo hàm thiết lập C() bởi vì hàm thiết lập sao chép đã được người sử dụng định nghĩa và đặt quyền truy xuất là **private**. Do vậy không thể dùng hàm thiết lập sao chép. Ví dụ:

C c;

Về lớp D:

Không thể khai báo đối tượng cho lớp D bởi vì trong lớp này chỉ có hàm thiết lập sao chép. Hàm thiết lập sao chép muốn sử dụng được thì phải có một đối tượng của lớp D. Do vậy muốn khai báo được một đối tượng thuộc lớp D thì trong lớp D cần có một hàm thiết lập khác để sao chép.

Ví dụ 5

Có bao nhiêu lần hàm thiết lập sao chép được gọi trong đoạn mã chương trình sau:

```
Widget f(Widget u)
{
    Widget v(u);
    Widget w=v;
    return w;
}

void main()
{
    Widget x;
    Widget y=f(f(x));
}
```

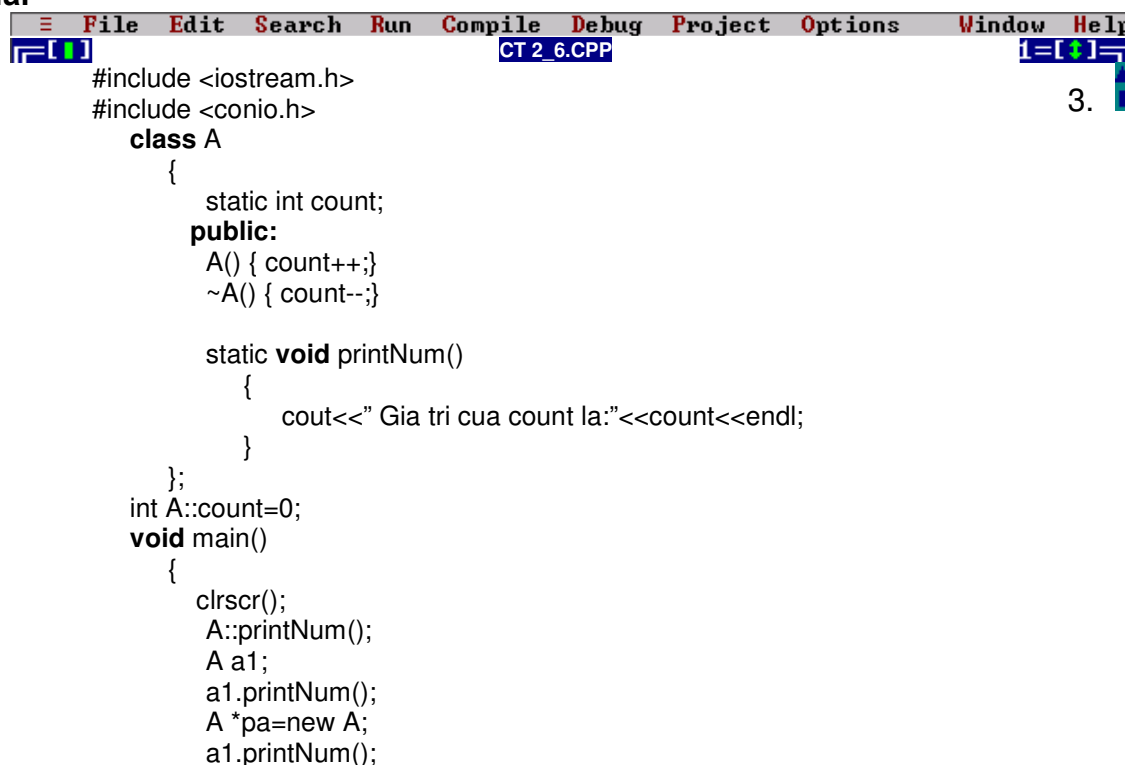
Lời giải:

Hàm thiết lập sao chép được gọi 7 lần trong đoạn mã chương trình này. Mỗi lần gọi hàm f đòi hỏi 3 lần gọi đến hàm thiết lập sao chép: khi tham số truyền vào bằng giá trị u, khi v và w được khởi tạo. Lệnh gọi thứ bảy là để khởi tạo y.

Ví dụ 6

Cho biết kết quả in ra màn hình của chương trình sau:

Lời giải



```

File Edit Search Run Compile Debug Project Options Window Help
CT 2_6.CPP
#include <iostream.h>
#include <conio.h>
class A
{
    static int count;
public:
    A() { count++;}
    ~A() { count--;}

    static void printNum()
    {
        cout<<" Gia tri cua count la:"<<count<<endl;
    }
};
int A::count=0;
void main()
{
    clrscr();
    A::printNum();
    A a1;
    a1.printNum();
    A *pa=new A;
    a1.printNum();
}
```

```

        delete pa;
        a1.printNum();
        A a2=a1;
        a2.printNum();
        getch();
    }

```

4. 

Lời giải

Kết quả in ra màn hình:

Gia trị của count là 0

Gia trị của count là 1

Gia trị của count là 2

Gia trị của count là 1

Gia trị của count là 1

Trong lớp A, thuộc tính count và hàm thành phần printNum là các thành phần tĩnh được chia sẻ bởi mọi đối tượng của A. Ban đầu thuộc tính count được khai báo khởi tạo là 0 (dòng `int A::count=0`), do vậy dòng đầu tiên của chương trình chính được gọi đến phương thức printNum sẽ đưa ra màn hình giá trị của count là 0. Tiếp theo ta tạo một đối tượng a1 sử dụng hàm thiết lập dạng `A()`. Hàm này làm tăng count lên 1. Do vậy, dòng gọi phương thức printNum tiếp theo sẽ đưa ra màn hình giá trị của count là 1. Tương tự đối với lệnh `new` ta cũng tạo ra một đối tượng mới và lúc này giá trị của count là 2. Sau khi sử dụng lệnh `new` tạo đối tượng, chương trình sử dụng `delete` để xóa đối tượng đó khỏi bộ nhớ. Lúc này hàm hủy bỏ được gọi và count giảm xuống 1. Đối tượng a2 được khai báo sử dụng hàm thiết lập sao chép mặc định, mà hàm này không làm thay đổi giá trị count. Do vậy, count vẫn giữ giá trị 1.

Ví dụ 6

Tìm ra chỗ sai về quyền truy xuất trong đoạn chương trình sau:

```

class A
{
    int pri;
    public:
    int pub;
    friend void funcA(A);
    friend class B;
};
class B
{
    void func(A a)
    {
        cout<<a.pri;
        cout<<a.pub;
    }
};
class C
{
    void func(A a)
    {
        cout<<a.pri;
        cout<<a.pub;
    }
};

```

```

    }
};
void funcA(A a)
{
    cout<<a.pri;
    cout<<a.pub;
}
void funcB(A a)
{
    cout<<a.pri;
    cout<<a.pub;
}

```

Lời giải

Truy xuất đến thuộc tính **pri** của đối tượng **a** trong hàm **funcB** và trong hàm thành phần **func** của lớp **C** là không thể được. Tất cả các truy xuất đến thuộc tính **pub** đều được bởi vì đây là thuộc tính **public**. Trong hàm **funcA**, ta có thể truy nhập được thuộc tính **pri** bởi vì hàm này đã được khai báo là bạn bè của lớp **A**. Tương tự, tất cả các hàm thành phần của lớp **B** cũng đều có thể truy xuất tới thuộc tính **pri** của lớp **A** bởi vì lớp **B** đã được coi là bạn bè của lớp **A**.

Ví dụ 7

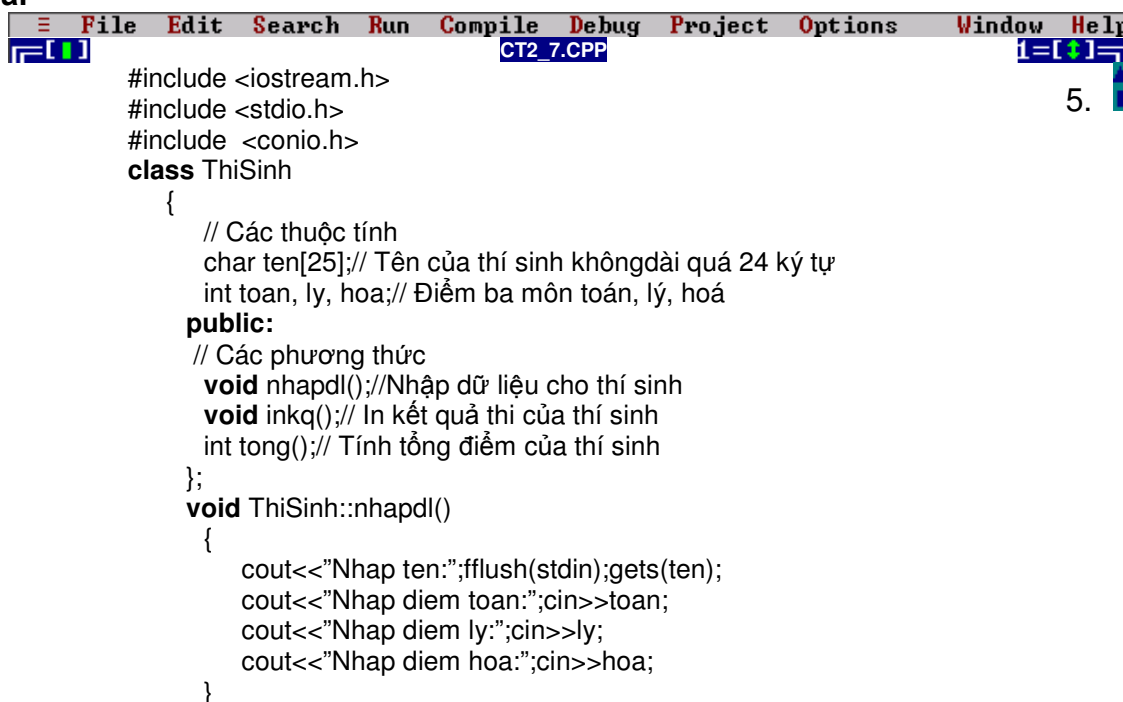
Để quản lý điểm thi vào trường ĐHSPKTHY của các thí sinh, ta xây dựng lớp **ThiSinh** mô tả các thí sinh bao gồm các thuộc tính và phương thức sau:

- Tên thí sinh
- Điểm của ba môn thi Toán, Lý, Hoá
- Nhập thông tin của các thí sinh gồm tên và điểm của ba môn thi Toán, Lý, Hoá
- In thông tin tên, điểm và tổng điểm thi 3 môn
- Tính tổng điểm thi của thí sinh

Trên cơ sở lớp đã xây dựng được, viết chương trình làm các công việc sau.

- Nhập danh sách kết quả thi của các thí sinh vào từ bàn phím
- Đưa ra màn hình danh sách thí sinh trung tuyển(điểm chuẩn vào trường là 18)

Lời giải



```

#include <iostream.h>
#include <stdio.h>
#include <conio.h>
class ThiSinh
{
    // Các thuộc tính
    char ten[25]; // Tên của thí sinh không dài quá 24 ký tự
    int toan, ly, hoa; // Điểm ba môn toán, lý, hoá
public:
    // Các phương thức
    void nhapdl(); // Nhập dữ liệu cho thí sinh
    void inkq(); // In kết quả thi của thí sinh
    int tong(); // Tính tổng điểm của thí sinh
};
void ThiSinh::nhapdl()
{
    cout<<"Nhập ten:"; fflush(stdin); gets(ten);
    cout<<"Nhập diem toan:"; cin>>toan;
    cout<<"Nhập diem ly:"; cin>>ly;
    cout<<"Nhập diem hoa:"; cin>>hoa;
}

```

```

void ThiSinh::inkq()
{ // Kết quả thi của thí sinh được in trên một dòng theo định dạng
  // Ten      Toan      Ly      Hoa
  Tong
  printf("%-25s%6d%6d%6d%6d\n",ten,toan,ly,hoa,tong());
}
int ThiSinh::tong()
{
  return (toan+ly+hoa);
}
void main()
{ // Chương trình chính thực hiện nhập danh sách vào số lượng thí
// sinh cần nhập
  clrscr();
  int n;
  cout<<"Cho số thí sinh:";cin>>n;
  // Tạo n đối tượng thí sinh cho n thí sinh cần nhập dữ liệu
  ThiSinh *dsts=new ThiSinh[n];
  // Nhập dữ liệu cho từng thí sinh
  for(int i=0;i<n;++i)
  {
    cout<<"Nhập dữ liệu cho thí sinh thứ:"<<i+1<<endl;
    // Gọi phương thức nhập dữ liệu của thí sinh thứ i trong mảng
    dsts[i].nhapdl();
  }
  // In danh sách các thí sinh trúng tuyển
  cout<<"Danh sách những người trúng tuyển \n";
  printf("%-25s%6s%6s%6s%6s\n","Ten","Toan","Ly","Hoa","Tong");
  for(i=0;i<n;++i)
    if(dsts[i].tong()>=18)
      dsts[i].inkq();
  // Xóa các đối tượng đã tạo và kết thúc chương trình
  delete dsts;
  getch();
}

```



6.

Ví dụ 8: Hàm thiết lập với các tham số mặc định

```

File Edit Search Run Compile Debug Project Options Window Help
CT2_8.CPP
#include <iostream.H>
class Time
{
public:
    Time(int = 0, int = 0, int = 0); //Constructor mặc định
    void SetTime(int, int, int);
    void PrintMilitary();
    void PrintStandard();

private:
    int Hour;
    int Minute;
    int Second;
};

//Hàm constructor để khởi tạo dữ liệu private
//Các giá trị mặc định là 0
Time::Time(int Hr, int Min, int Sec)
{

```

7.

```

        SetTime(Hr, Min, Sec);
    }

    //Thiet lap cac gia tri cua Hour, Minute va Second
    //Gia tri khong hop le duoc thiet lap la 0
    void Time::SetTime(int H, int M, int S)
    {
        Hour = (H >= 0 && H < 24) ? H : 0;
        Minute = (M >= 0 && M < 60) ? M : 0;
        Second = (S >= 0 && S < 60) ? S : 0;
    }

    //Hien thi thoi gian theo dang gio quan doi: HH:MM:SS
    void Time::PrintMilitary()
    {
        cout << (Hour < 10 ? "0" : "") << Hour << ":"
            << (Minute < 10 ? "0" : "") << Minute << ":"
            << (Second < 10 ? "0" : "") << Second;
    }

    //Hien thi thoi gian theo dang chuan: HH:MM:SS AM (hoac PM)
    void Time::PrintStandard()
    {
        cout << ((Hour == 0 || Hour == 12) ? 12 : Hour % 12)
            << ":" << (Minute < 10 ? "0" : "") << Minute
            << ":" << (Second < 10 ? "0" : "") << Second
            << (Hour < 12 ? " AM" : " PM");
    }

    int main()
    {
        Time T1,T2(2),T3(21,34),T4(12,25,42),T5(27,74,99);

        cout << "Constructed with:" << endl
            << "all arguments defaulted:" << endl << " ";
        T1.PrintMilitary();
        cout << endl << " ";
        T1.PrintStandard();
        cout << endl << "Hour specified; Minute and Second defaulted:"
            << endl << " ";
        T2.PrintMilitary();
        cout << endl << " ";
        T2.PrintStandard();
        cout << endl << "Hour and Minute specified; Second defaulted:"
            << endl << " ";
        T3.PrintMilitary();
        cout << endl << " ";
        T3.PrintStandard();
        cout << endl << "Hour, Minute, and Second specified:"
            << endl << " ";
        T4.PrintMilitary();
        cout << endl << " ";
        T4.PrintStandard();
        cout << endl << "all invalid values specified:"
            << endl << " ";
        T5.PrintMilitary();
        cout << endl << " ";
        T5.PrintStandard();
        cout << endl;
        return 0;
    }

```




Chương trình ở ví dụ trên khởi tạo năm đối tượng của lớp *Time* (ở dòng 52). Đối tượng *T1* với ba tham số lấy giá trị mặc định, đối tượng *T2* với một tham số được mô tả, đối tượng *T3* với hai tham số được mô tả, đối tượng *T4* với ba tham số được mô tả và đối tượng *T5* với các tham số có giá trị không hợp lệ.

Kết quả

```
Constructed with:
all arguments defaulted:
    00:00:00
    12:00:00 AM
Hour specified; Minute and Second defaulted:
    02:00:00
    2:00:00 AM
Hour and Minute specified; Second defaulted:
    21:34:00
    9:34:00 PM
Hour, Minute, and Second specified:
    12:25:42
    12:25:42 PM
all invalid values specified:
    00:00:00
    12:00:00 AM
```

Ví dụ 9: Lớp có hàm hủy bỏ

```

File Edit Search Run Compile Debug Project Options Window Help
CT 2_9.CPP
9.
#include <iostream.h>
class Simple
{
    private:
        int *X;
    public:
        Simple();           //Constructor
        ~Simple();          //Destructor
        void SetValue(int V);
        int GetValue();
};

Simple::Simple()
{
    X = new int; //Cấp phát vùng nhớ cho X
}

Simple::~~Simple()
{
    delete X; //Giải phóng vùng nhớ khi đối tượng bị hủy bỏ
}

```

```

void Simple::SetValue(int V)
{
    *X = V;
}

int Simple::GetValue()
{
    return *X;
}

int main()
{
    Simple S;
    int X;

    cout<<"Enter a number:";
    cin>>X;
    S.SetValue(X);
    cout<<"The value of this number:"<<S.GetValue();

    return 0;
}

```



kết quả

```

Enter a number: 12
The value of this number: 12

```

Ví dụ 10: Chương trình sau minh họa khai báo và sử dụng hàm **friend**.

```

#include <iostream.H>
class Count
{
    friend void SetX(Count &, int); //Khai bao friend
public:
    Count() //Constructor
    {
        X = 0;
    }
    void Print() const //Xuat
    {
        cout << X << endl;
    }
private:
    int X;
};

//Co the thay doi du lieu private cua lop Count vi
//SetX() khai bao la mot ham friend cua lop Count
void SetX(Count &C, int Val)
{

```

```

        C.X = Val; //Hop le: SetX() la mot friend cua lop Count
    }

    int main()
    {
        Count Object;

        cout << "Object.X after instantiation: ";
        Object.Print();
        cout << "Object.X after call to SetX friend function: ";
        SetX(Object, 8); //Thiet lap X voi mot friend
        Object.Print();
        return 0;
    }

```

1:3

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

kết quả

```

Object.X after instantiation: 0
Object.X after call to SetX friend function: 8

```

D/ BÀI TẬP TỰ GIẢI

Câu hỏi trắc nghiệm

Câu 1:

Chỉ có một khẳng định trong những câu sau là sai. Câu nào?

- a./ Mỗi thể hiện của một lớp có sở hữu riêng các thuộc tính thông thường
- b./ Các thể hiện của một lớp cùng chia sẻ các thuộc tính tĩnh của lớp đó
- c./ Mỗi thể hiện của một lớp có các định nghĩa riêng cho các phương thức của nó.
- d./ Mỗi đối tượng là một thể hiện của một lớp

Câu 2:

Các từ khoá public và **private** dùng để

- a./ Cho phép người thiết kế lớp che dấu một phần thi hành của lớp trước người sử dụng lớp.
- b./ Trình biên dịch tối ưu hoá chương trình
- c./ Đảm bảo na toàn của lớp khi thiết kế
- d./ Hạn chế việc sao chép lớp

Câu 3

Chỉ ra lỗi sai với các khai báo cho lớp A

```

class A
{
    A(int i);
};

```

```

A a1;//(1)
A b2(5) //(2)

```

- a./ Chỉ dòng 1 lỗi
- b./ Chỉ dòng 2 lỗi
- c./ Cả hai dòng lỗi
- d./ Không dòng nào lỗi

Câu 4:

Cho biết giá trị của n với các dòng lệnh sau:

```
class A
{
    public:
        static int i;
};
int A::i=5;
A a1;
a1.i++;
A a2;
int n=a2.i+1;
a./ n=5;
b./ n=6;
c./ n=7;
d./ Không câu nào đúng
```

Câu 5:

Chỉ ra lỗi với các khai báo cho lớp A

```
class A
{
    public:
        A(int i);
};
A a1(5);
A a3;// (1)
A a2=a1;// (2)
a./ Chỉ dòng 1 lỗi
b./ Chỉ dòng 2 lỗi
c./ Cả 2 dòng lỗi
d./ Không dòng nào lỗi
```

Bài tập

Bài 2.1: Xây dựng lớp Stack, dữ liệu bao gồm đỉnh stack và vùng nhớ của stack. Các thao tác gồm:

- Khởi động stack.
- Kiểm tra stack có rỗng không?
- Kiểm tra stack có đầy không?
- Push và pop.

Bài 2.2: Xây dựng lớp hình trụ Cylinder, dữ liệu bao gồm bán kính và chiều cao của hình trụ. Các thao tác gồm hàm tính diện tích toàn phần và thể tích của hình trụ đó.

Bài 2.3: Hãy xây dựng một lớp Point cho các điểm trong không gian ba chiều (x,y,z). Lớp chứa một constructor mặc định, một hàm Negate() để biến đổi điểm thành đại lượng có dấu âm, một hàm Norm() trả về khoảng cách từ gốc và một hàm Print().

Bài 2.4: Xây dựng một lớp Matrix cho các ma trận bao gồm một constructor mặc định, hàm xuất ma trận, nhập ma trận từ bàn phím, cộng hai ma trận, trừ hai ma trận và nhân hai ma trận.

Bài 2.5: Xây dựng một lớp Matrix cho các ma trận vuông bao gồm một constructor mặc định, hàm xuất ma trận, tính định thức và tính ma trận nghịch đảo.

Bài 2.6: Xây dựng lớp Person để quản lý họ tên, năm sinh, điểm chín môn học của tất cả các học viên của lớp học. Cho biết bao nhiêu học viên trong lớp được phép làm luận văn tốt nghiệp, bao nhiêu học viên thi tốt nghiệp, bao nhiêu người phải thi lại và tên môn thi lại. Tiêu chuẩn để xét:

- Làm luận văn phải có điểm trung bình lớn hơn 7 trong đó không có môn nào dưới 5.
- Thi tốt nghiệp khi điểm trung bình không lớn hơn 7 và điểm các môn không dưới 5.
- Thi lại có môn dưới 5.

Bài 2.7: Xây dựng một lớp String. Mỗi đối tượng của lớp String sẽ đại diện một chuỗi ký tự. Các thành viên dữ liệu là chiều dài chuỗi và chuỗi ký tự thực. Ngoài constructor và destructor còn có các phương thức như tạo một chuỗi với chiều dài cho trước, tạo một chuỗi từ một chuỗi đã có.

Bài 2.8: Xây dựng một lớp Vector để lưu trữ vector gồm các số thực. Các thành viên dữ liệu gồm:

- Kích thước vector.
- Một mảng động chứa các thành phần của vector.

Ngoài constructor và destructor, còn có các phương thức tính tích vô hướng của hai vector, tính chuẩn của vector (theo chuẩn bất kỳ nào đó).

Bài 2.9: Xây dựng lớp Employee gồm họ tên và chứng minh nhân dân. Ngoài constructor còn có phương thức nhập, xuất họ tên và chứng minh nhân dân ra màn hình

Bài 2.10:

Một lớp đối tượng sách trong hệ thống quản lí thư viện có các thuộc tính

- Tên sách
- Tổng số quyền sách
- Số quyền sách đã cho mượn

Xây dựng lớp đối tượng trên với các phương thức như sau

- Phương thức nhập dữ liệu cho đối tượng từ bàn phím. Các thông tin cần nhập là tên sách, tổng số sách, số đã cho mượn.
- Phương thức in thông tin đối tượng ra màn hình bao gồm tên, tổng số và số đã cho mượn.
- Phương thức tính số sách còn lại trong thư viện (tổng số - số mượn)

Trên cơ sở lớp xây dựng được, viết chương trình chính thực hiện các công việc.

- Nhập danh sách các quyền sách với số lượng sách cần nhập được cho vào từ bàn phím.
- Đưa ra màn hình thông tin về các quyền sách hiện có trong thư viện (số sách còn lại phải lớn hơn 0)

Bài 2.11:

Viết một lớp biểu diễn hình chữ nhật có các thuộc tính là độ dài hai cạnh(chiều rộng và chiều dài) và có các phương thức sau.

- Nhập dữ liệu hai cạnh cho hình chữ nhật
- Tính chu vi và diện tích hình chữ nhật
- In thông tin của hình chữ nhật ra màn hình(bao gồm độ dài hai cạnh, chu vi và diện tích)

Trên cơ sở lớp xây dựng được viết chương trình cho phép người sử dụng nhập dữ liệu của một hình chữ nhật rồi in thông tin về nó ra màn hình.

Bài 2.12:

Xây dựng một lớp Date mô tả thông tin về ngày, tháng, năm(month, day, year). Lớp Date có các hàm thành phần:

- Hàm thiết lập với ba tham số có giá trị mặc định(đó là ngày hệ thống)
- Nhập dữ liệu ngày, tháng và năm
- Hàm in thông tin về ngày tháng năm dưới dạng mm-dd-yy
- Hàm nextDay() để tăng Date từng ngày một

Trên cơ sở lớp Date vừa xây dựng viết chương trình cho biết khoảng cách ngày giữa hai mốc thời gian với ngày bắt đầu được nhập vào từ bàn phím cho tới ngày hiện thời.

Bài 2.13:

Xây dựng lớp Stack và lớp Queue mô tả hoạt động của ngăn xếp và hàng đợi các số nguyên.

Bài 2.14:

Xây dựng một lớp mô tả các bảng thi đấu bóng đá gọi là BangThiDau. Giả thiết mỗi bảng có bốn đội và thi đấu chéo từng cặp. Có lịch các trận đấu của bảng. Tạo các phương thức nhập kết quả thi đấu và tính điểm cho từng đội. Yêu cầu việc nhập kết quả thi đấu phải theo thứ tự thời gian. Thêm các phương thức hiện thị thông tin thi đấu của từng đội và của cả bảng. Viết chương trình để kiểm nghiệm lớp xây dựng được.

Bài 2.15:

Mở rộng bài tập trên với lớp DoiBong mô tả các đội bóng thi đấu. Thông tin của mỗi đội bóng bao gồm tên đội bóng, danh sách cầu thủ, và huấn luyện viên. Lớp bangThiDau sử dụng các đối tượng của lớp DoiBong để làm đội bóng thi đấu của bảng. Các bảng lúc này có thêm thuộc tính tên của bảng thi đấu.

Sử dụng các lớp đã xây dựng ở trên để viết chương trình quản lý thi đấu của cúp bóng đá thế giới có 32 đội thi đấu được chia làm 8 bảng. Tạo thêm lớp mô tả lịch thi đấu cho các vòng tiếp theo của giải. Yêu cầu của chương trình như sau:

- Ban đầu người sử dụng nhập các thông tin về đội bóng, sau đó phân bảng và lịch thi đấu toàn giải.
- Kết quả các trận thi đấu được vào theo thứ tự lịch thi đấu
- Chương trình tự động chọn các đội vào vòng tiếp theo cho tới trận trung kết.
- Tại mỗi thời điểm của giải chương trình có thể đưa ra các thông tin về giải.

Bài 2.16:

Một quyển sổ điện thoại chứa các thẻ có thông tin về tên, địa chỉ và số điện thoại. Thiết kế các lớp tương ứng với các thẻ thông tin và số điện thoại. Viết chương trình cho phép quản lý số điện thoại dựa trên các lớp xây dựng được.

ĐỊNH NGHĨA CHỒNG TOÁN TỬ TRÊN LỚP

MỤC TIÊU CỦA BÀI NÀY GIÚP NGƯỜI HỌC

- Cách định nghĩa các phép toán cho kiểu dữ liệu lớp và cấu trúc
- Các toán tử chuyển kiểu áp dụng cho kiểu dữ liệu lớp

A/ NHẮC LẠI LÝ THUYẾT

Toán tử được định nghĩa chồng bằng cách định nghĩa một hàm toán tử. Tên hàm toán tử bao gồm từ khoá **operator** theo sau là ký hiệu của toán tử được định nghĩa chồng.

Hầu hết các toán tử của C++ đều có thể định nghĩa chồng. Không thể tạo ra các ký hiệu phép toán mới.

Phải đảm bảo các đặc tính nguyên thủy của toán tử được định nghĩa chồng, chẳng hạn: độ ưu tiên, trật tự kết hợp, số ngôi.

Không sử dụng tham số có giá trị ngầm định để định nghĩa chồng toán tử.

Các toán tử `()`, `[]`, `->`, `=` yêu cầu hàm toán tử phải là hàm thành phần của lớp.

Hàm toán tử có thể là hàm thành phần hay là hàm bạn của lớp

Khi hàm toán tử là hàm thành phần, toán hạng bên trái luôn là đối tượng thuộc lớp.

Nếu toán hạng bên trái là đối tượng của lớp khác thì hàm toán tử tương ứng phải là hàm bạn.

Chương trình dịch không tự biết cách chuyển kiểu giữa kiểu dữ liệu chuẩn và kiểu dữ liệu tự định nghĩa. Vì vậy người lập trình cần phải mô tả tường minh các chuyển đổi này dưới dạng hàm thiết lập chuyển kiểu hay hàm toán tử chuyển kiểu.

Một hàm toán tử chuyển kiểu thực hiện chuyển đổi từ một đối tượng thuộc lớp sang đối tượng thuộc lớp khác hoặc một đối tượng có kiểu được định nghĩa trước.

Hàm thiết lập chuyển kiểu có một tham số và thực hiện chuyển đổi từ một giá trị sang đối tượng kiểu lớp.

Toán tử gán là toán tử hay được định nghĩa chồng nhất, đặc biệt khi lớp có các thành phần dữ liệu động.

Để định nghĩa chồng toán tử tăng, giảm một ngôi, phải phân biệt hai hàm toán tử tương ứng cho dạng tiền tố và dạng hậu tố.

B. MỘT SỐ LƯU Ý (Các lỗi thường gặp, một số thói quen lập trình tốt...)

Các lỗi thường gặp

- Tạo một toán tử
- Thay đổi định nghĩa của các toán tử trên các kiểu được định nghĩa trước
- Cho rằng việc định nghĩa chồng một toán tử sẽ tự động kéo theo định nghĩa chồng của các toán tử liên quan.
- Quên định nghĩa chồng toán tử gán và hàm thiết lập sao chép cho các lớp có các thành phần dữ liệu động.

Một số thói quen lập trình tốt

- Sử dụng toán tử định nghĩa chồng khi điều đó làm cho chương trình trong sáng hơn.
- Tránh lạm dụng định nghĩa chồng toán tử vì điều đó dẫn đến khó kiểm soát chương trình.
- Chú ý đến các tính chất nguyên thủy của toán tử được định nghĩa chồng.
- Hàm thiết lập, toán tử gán, hàm thiết lập sao chép của một lớp thường đi cùng nhau.

C/ BÀI TẬP MẪU**Ví dụ 1:**

Một lớp phân số có toán tử cộng(+) được định nghĩa như sau:

```
class PS
{
    public:
        PS(int ts=0, int ms=1);
        PS operator+(PS);
};
```

Trong các dòng lệnh sau đây dòng nào sai?

```
PS a,b,c;
a=b+c;//(1)
a=b+3;//(2)
a=3+b;//(3)
```

Lời giải

Trong ba dòng lệnh thì hai dòng đầu là đúng bởi vì lúc đó ta có:

(1) $\Leftrightarrow a=b.operator+(c)$ là toán tử đã được định nghĩa trong lớp phân số

(2) $\Leftrightarrow a=b.operator+(3)$ với 3 sẽ tự động chuyển kiểu thành phân số

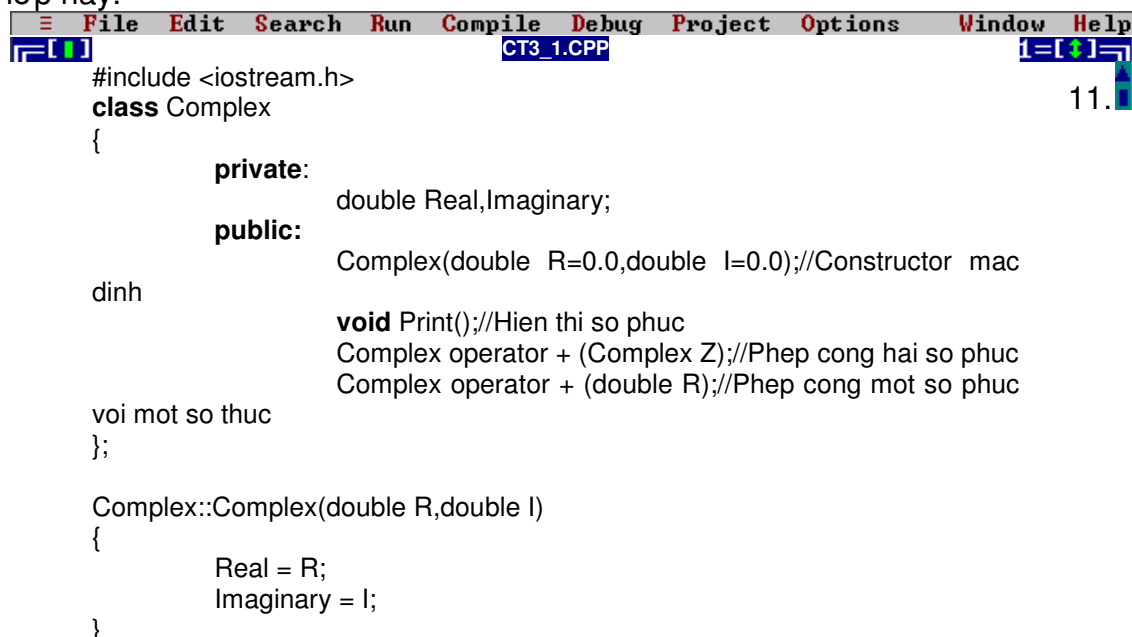
Dòng lệnh (3) sai vì ta không có toán tử cộng một số nguyên với một phân số. Để có thể thực hiện được tất cả ba dòng lệnh như trên thì toán tử cộng trong lớp PS phải được định nghĩa là một hàm bạn.

```
class PS
{
    public:
        PS(int ts=0, int ms=1);
        friend PS operator+(PS);
};
```

Khi đó các lời gọi sẽ tương ứng với toán tử như sau:

```
(1)  $\Leftrightarrow a=operator+(b,c)$ 
(2)  $\Leftrightarrow a=operator+(b,3)$ 
(3)  $\Leftrightarrow a=operator+(3,b)$ 
```

Ví dụ 2: Chúng ta xây dựng lớp số phức với tên lớp là *Complex* và đa năng hóa toán tử + trên lớp này.



```
File Edit Search Run Compile Debug Project Options Window Help
CT3_1.CPP 11.
#include <iostream.h>
class Complex
{
    private:
        double Real,Imaginary;
    public:
        Complex(double R=0.0,double I=0.0);//Constructor mac
        void Print();//Hien thi so phuc
        Complex operator + (Complex Z);//Phep cong hai so phuc
        Complex operator + (double R);//Phep cong mot so phuc
};

Complex::Complex(double R,double I)
{
    Real = R;
    Imaginary = I;
}
```



```

void Complex::Print()
{
    cout<<'('<<Real<<','<<Imaginary<<')';
}

Complex Complex::operator + (Complex Z)
{
    Complex Tmp;
    Tmp.Real = Real + Z.Real;
    Tmp.Imaginary = Imaginary + Z.Imaginary;
    return Tmp;
}

Complex Complex::operator + (double R)
{
    Complex Tmp;
    Tmp.Real = Real + R;
    Tmp.Imaginary = Imaginary;
    return Tmp;
}

int main()
{
    Complex X,Y(4.3,8.2),Z(3.3,1.1);
    cout<<"X: ";
    X.Print();
    cout<<endl<<"Y: ";
    Y.Print();
    cout<<endl<<"Z: ";
    Z.Print();
    X = Y + Z;
    cout<<endl<<endl<<"X = Y + Z:"<<endl;
    X.Print();
    cout<<" = ";
    Y.Print();
    cout<<" + ";
    Z.Print();
    X = Y + 3.5;
    cout<<endl<<endl<<"X = Y + 3.5:"<<endl;
    X.Print();
    cout<<" = ";
    Y.Print();
    cout<<" + 3.5";
    return 0;
}

```



kết quả

```

X: (0,0)
Y: (4.3,8.2)
Z: (3.3,1.1)

X = Y + Z:
(7.6,9.3) = (4.3,8.2) + (3.3,1.1)

X = Y + 3.5:
(7.8,8.2) = (4.3,8.2) + 3.5

```

Ví dụ 3:

Cho một lớp có toán tử chuyển kiểu như sau:

```

class A
{
    public:
        operator int ();
};

```

Khi đó ta có thể sử dụng câu lệnh nào trong những câu lệnh sau:

```

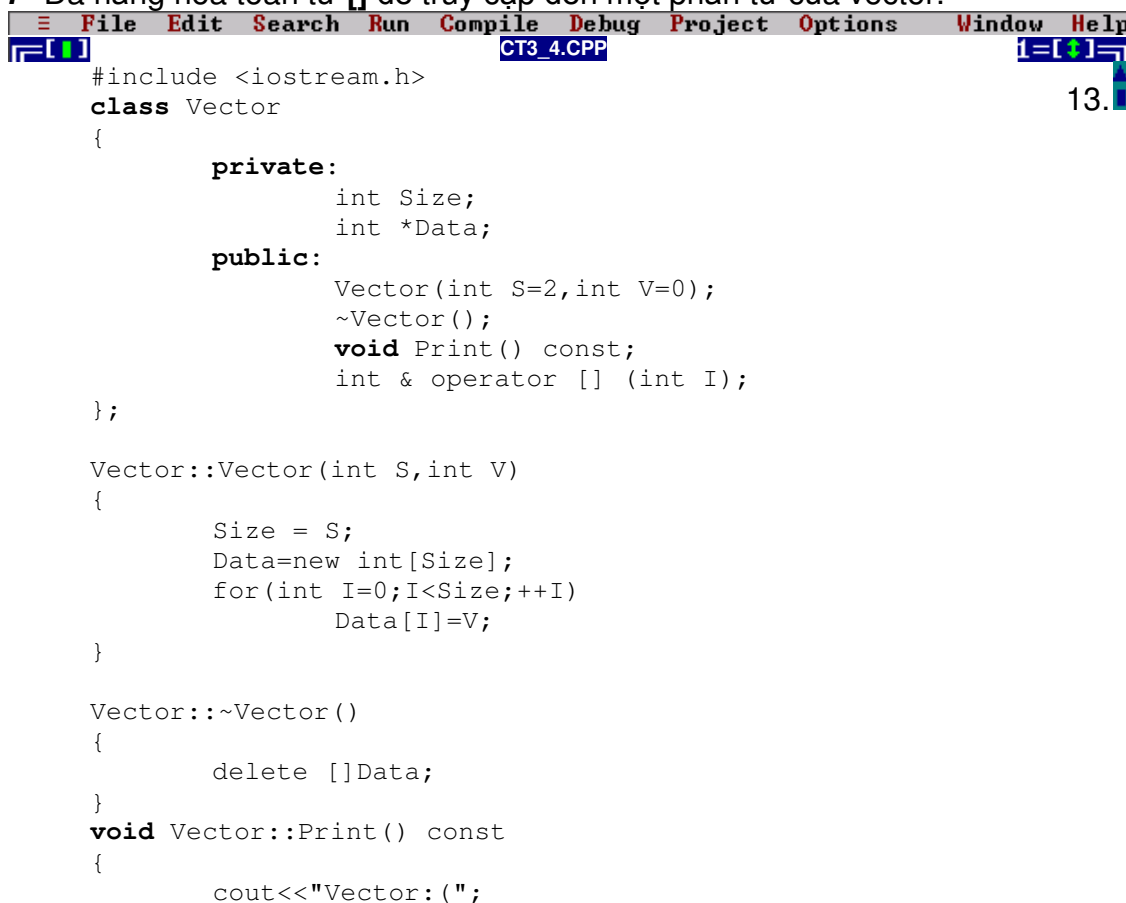
A a;
int i=a;//(1)
float f=a;//(2)

```

Lời giải

Lệnh (1) là đúng bởi vì ta đã có toán tử ép kiểu int nên chương trình tự động ép kiểu về một số nguyên. Còn lệnh 2 sai bởi vì ta không có toán tử ép kiểu float.

Ví dụ 4 Đa năng hóa toán tử [] để truy cập đến một phần tử của vector.



```

File Edit Search Run Compile Debug Project Options Window Help
CT3_4.CPP
#include <iostream.h>
class Vector
{
    private:
        int Size;
        int *Data;
    public:
        Vector(int S=2,int V=0);
        ~Vector();
        void Print() const;
        int & operator [] (int I);
};

Vector::Vector(int S,int V)
{
    Size = S;
    Data=new int[Size];
    for(int I=0;I<Size;++I)
        Data[I]=V;
}

Vector::~~Vector()
{
    delete []Data;
}

void Vector::Print() const
{
    cout<<"Vector: (";

```

```

        for(int I=0;I<Size-1;++I)
            cout<<Data[I]<<" ";
        cout<<Data[Size-1]<<" "<<endl;
    }

    int & Vector::operator [] (int I)
    {
        return Data[I];
    }

    int main()
    {
        Vector V(5,1);
        V.Print();
        for(int I=0;I<5;++I)
            V[I]*=(I+1);
        V.Print();
        V[0]=10;
        V.Print();
        return 0;
    }

```



14.

kết quả

```

Vector:(1,1,1,1,1)
Vector:(1,2,3,4,5)
Vector:(10,2,3,4,5)

```

Ví dụ 5: Đa năng hóa toán tử `()` để truy cập đến một phần tử của vector.

```

//Chương trình 4.6
#include <iostream.h>

class Vector
{
    private:
        int Size;
        int *Data;
    public:
        Vector(int S=2,int V=0);
        ~Vector();
        void Print() const;
        int & operator () (int I);
};

Vector::Vector(int S,int V)
{
    Size = S;
    Data=new int[Size];
    for(int I=0;I<Size;++I)
        Data[I]=V;
}

Vector::~~Vector()
{
}

```

15.

```

        delete []Data;
    }
    void Vector::Print() const
    {
        cout<<"Vector:(";
        for(int l=0;l<Size-1;++l)
            cout<<Data[l]<<" ";
        cout<<Data[Size-1]<<")"<<endl;
    }

    int & Vector::operator()(int l)
    {
        return Data[l];
    }

    int main()
    {
        Vector V(5,1);
        V.Print();
        for(int l=0;l<5;++l)
            V(l)*=(l+1);

        V.Print();
        V(0)=10;
        V.Print();
        return 0;
    }

```

16. 

kết

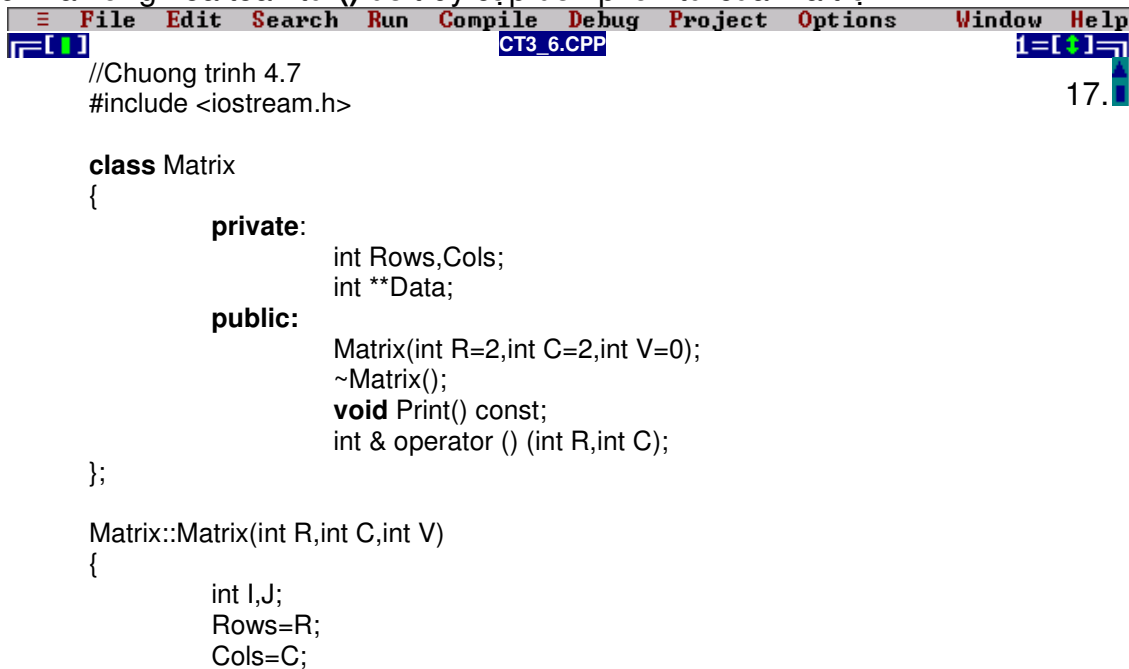
quả

```

Vector:(1,1,1,1,1)
Vector:(1,2,3,4,5)
Vector:(10,2,3,4,5)

```

Ví dụ 6: Đa năng hóa toán tử () để truy cập đến phần tử của ma trận.

17. 

```

//Chương trình 4.7
#include <iostream.h>

class Matrix
{
    private:
        int Rows,Cols;
        int **Data;

    public:
        Matrix(int R=2,int C=2,int V=0);
        ~Matrix();
        void Print() const;
        int & operator () (int R,int C);
};

Matrix::Matrix(int R,int C,int V)
{
    int I,J;
    Rows=R;
    Cols=C;

```

```

        Data = new int *[Rows];
        int *Temp=new int[Rows*Cols];
        for(l=0;l<Rows;++l)
        {
            Data[l]=Temp;
            Temp+=Cols;
        }
        for(l=0;l<Rows;++l)
            for(J=0;J<Cols;++J)
                Data[l][J]=V;
    }

    Matrix::~Matrix()
    {
        delete [] Data[0];
        delete [] Data;
    }

    void Matrix::Print() const
    {
        int l,J;
        for(l=0;l<Rows;++l)
        {
            for(J=0;J<Cols;++J)
            {
                cout.width(5);//Hien thi canh ler phai voi chieu
                cout<<Data[l][J];
            }
            cout<<endl;
        }
    }

    int & Matrix::operator () (int R,int C)
    {
        return Data[R][C];
    }

    int main()
    {
        int l,J;
        Matrix M(2,3,1);
        cout<<"Matrix:"<<endl;
        M.Print();
        for(l=0;l<2;++l)
            for(J=0;J<3;++J)
                M(l,J)=(l+J+1);
        cout<<"Matrix:"<<endl;
        M.Print();
        return 0;
    }

```



kết quả

Matrix:

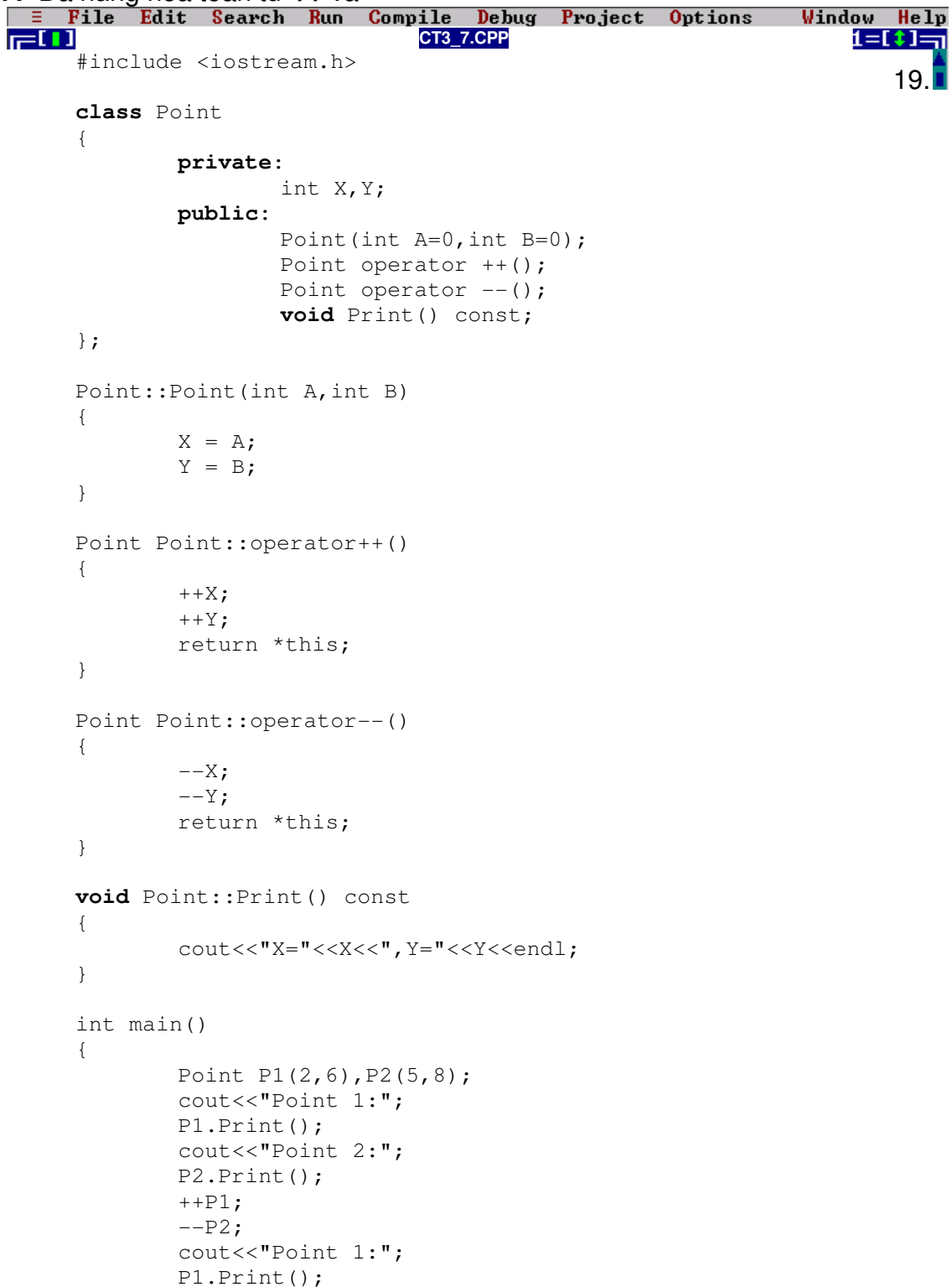
1 1 1

1 1 1

Matrix:

1 2 3

2 3 4

Ví dụ 7: Đa năng hóa toán tử ++ và --


```

#include <iostream.h>

class Point
{
    private:
        int X,Y;
    public:
        Point(int A=0,int B=0);
        Point operator ++();
        Point operator --();
        void Print() const;
};

Point::Point(int A,int B)
{
    X = A;
    Y = B;
}

Point Point::operator++()
{
    ++X;
    ++Y;
    return *this;
}

Point Point::operator--()
{
    --X;
    --Y;
    return *this;
}

void Point::Print() const
{
    cout<<"X="<<X<<" ,Y="<<Y<<endl;
}

int main()
{
    Point P1(2,6),P2(5,8);
    cout<<"Point 1:";
    P1.Print();
    cout<<"Point 2:";
    P2.Print();
    ++P1;
    --P2;
    cout<<"Point 1:";
    P1.Print();
}

```

```

        cout<<"Point 2:";
        P2.Print();
        return 0;
    }

```



20.

Kết quả

```

Point 1:X=2,Y=6
Point 2:X=5,Y=8
Point 1:X=3,Y=7
Point 2:X=4,Y=7

```

Ví dụ 8: Đa năng hóa toán tử dấu phẩy.



```

#include <iostream.h>

class Point
{
    private:
        int X,Y;
    public:
        Point(int A=0,int B=0);
        Point operator +(Point P);
        Point operator ,(Point P);
        void Print() const;
};

Point::Point(int A,int B)
{
    X = A;
    Y = B;
}

Point Point::operator+(Point P)
{
    Point Tmp;
    Tmp.X=X+P.X;
    Tmp.Y=Y+P.Y;
    return Tmp;
}

Point Point::operator,(Point P)
{
    Point Tmp;
    Tmp.X=P.X;
    Tmp.Y=P.Y;
    cout<<P.X<<" "<<P.Y<<endl;
    return Tmp;
}

void Point::Print() const
{
    cout<<"X="<<X<<" ,Y="<<Y<<endl;
}

```

21.

```

int main()
{
    Point P1(2,6), P2(5,20), P3(1,1);
    cout<<"Point 1:";
    P1.Print();
    cout<<"Point 2:";
    P2.Print();
    cout<<"Point 3:";
    P3.Print();
    P1=(P1,P1+P2,P3);
    cout<<"Point 1:";
    P1.Print();
    return 0;
}

```

22.



kết quả

```

Point 1:X=2,Y=6
Point 2:X=5,Y=20
Point 3:X=1,Y=1
7 26
1 1
Point 1:X=1,Y=1

```

Ví dụ 9: Đa năng hóa toán tử ->.

```

#include <iostream.h>

class MyClass
{
public:
    int Data;
    MyClass * operator ->()
    {
        return this;
    }
};

int main()
{
    MyClass M;
    M->Data = 10;
    cout<<M.Data<<" "<<M->Data;
    return 0;
}

```

23.



24.

kết quả

```

10 10

```

Ví dụ 10: Đa năng hóa toán tử gán.


```

#include <iostream.h>
#include <string.h>

class String
{
    private:
        char *St;
        int Len;
    public:
        String(char *S);
        ~String();
        char *GetStr();
        String & operator=(String &Obj);
};

String::String(char *S)
{
    Len=strlen(S);
    St=new char[Len+1];
    strcpy(St,S);
}

String::~~String()
{
    delete []St;
}

char * String::GetStr()
{
    return St;
}

String & String::operator=(String &Obj)
{
    if (Len< Obj.Len)
    {
        delete [] St;
        St=new char[Obj.Len+1];
    }
    Len=Obj.Len;
    strcpy(St,Obj.St);
    return *this;
}

int main()
{
    String S1("Hello"), S2("Chao");
    cout<<"S1="<<S1.GetStr()<<endl;
    cout<<"S2="<<S2.GetStr()<<endl;
    S2=S1;
    cout<<"S1="<<S1.GetStr()<<endl;
    cout<<"S2="<<S2.GetStr()<<endl;
    return 0;
}

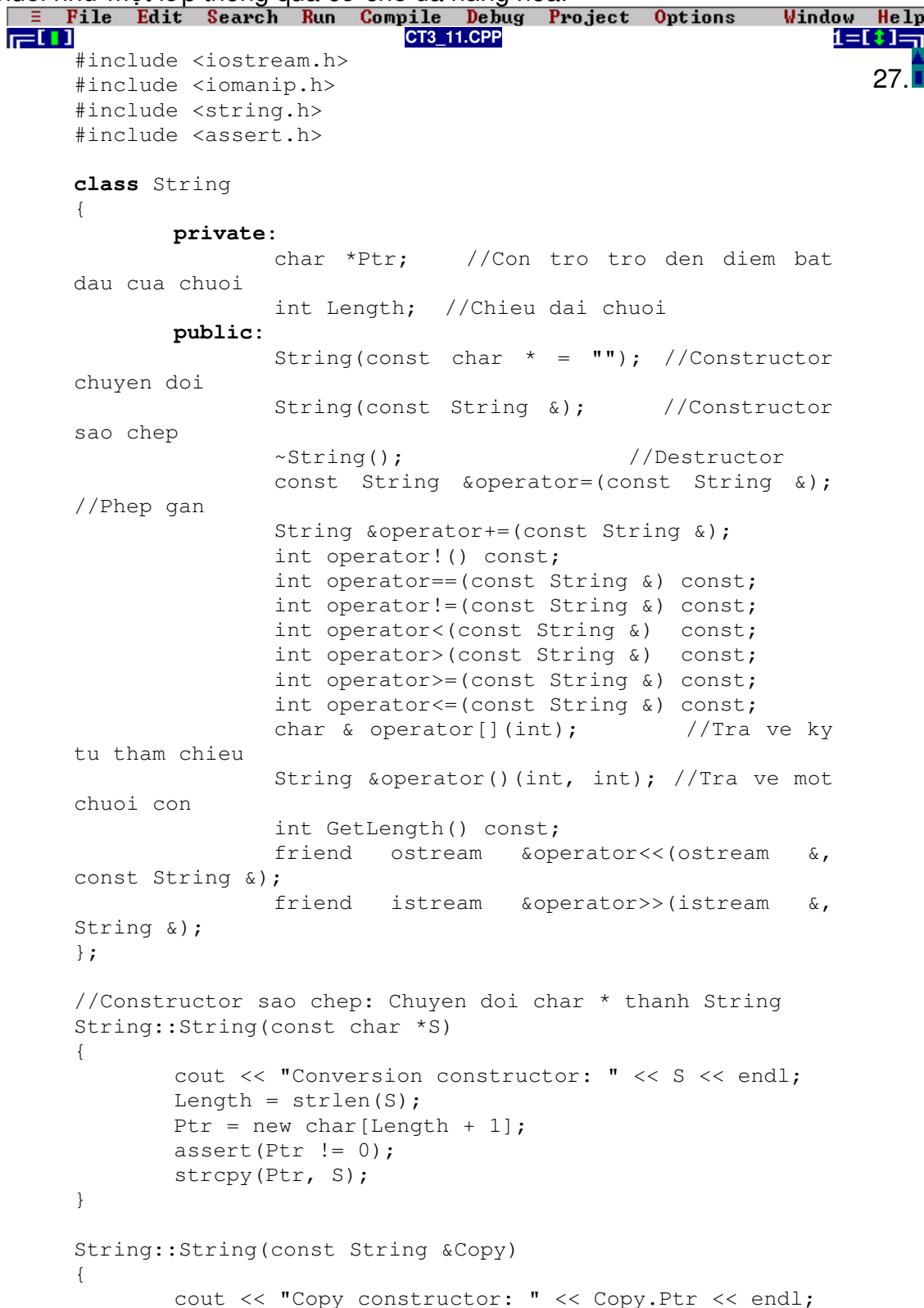
```



kết quả

```
S1=Hello
S2=Chao
S1=Hello
S2=Hello
```

Ví dụ 11: Chúng ta sẽ xây dựng một lớp xử lý việc tạo và thao tác trên các chuỗi (string). C++ không cài sẵn kiểu dữ liệu chuỗi. Nhưng C++ cho phép chúng ta thêm kiểu chuỗi như một lớp thông qua cơ chế đa năng hóa.



```
#include <iostream.h>
#include <iomanip.h>
#include <string.h>
#include <assert.h>

class String
{
    private:
        char *Ptr;    //Con tro tro den diem bat
        dau cua chuoi
        int Length;   //Chieu dai chuoi
    public:
        String(const char * = ""); //Constructor
        String(const String &);    //Constructor
        ~String();                //Destructor
        const String &operator=(const String &);

        //Phep gan
        String &operator+=(const String &);
        int operator!() const;
        int operator==(const String &) const;
        int operator!=(const String &) const;
        int operator<(const String &) const;
        int operator>(const String &) const;
        int operator>=(const String &) const;
        int operator<=(const String &) const;
        char &operator[](int);    //Tra ve ky
        tu tham chieu
        String &operator()(int, int); //Tra ve mot
        chuoi con
        int GetLength() const;
        friend ostream &operator<<(ostream &,
        const String &);
        friend istream &operator>>(istream &,
        String &);
};

//Constructor sao chep: Chuyen doi char * thanh String
String::String(const char *S)
{
    cout << "Conversion constructor: " << S << endl;
    Length = strlen(S);
    Ptr = new char[Length + 1];
    assert(Ptr != 0);
    strcpy(Ptr, S);
}

String::String(const String &Copy)
{
    cout << "Copy constructor: " << Copy.Ptr << endl;
```

```
        Length = Copy.Length;
        Ptr = new char[Length + 1];
        assert(Ptr != 0);
        strcpy(Ptr, Copy.Ptr);
    }

    //Destructor
    String::~String()
    {
        cout << "Destructor: " << Ptr << endl;
        delete [] Ptr;
    }

    const String &String::operator=(const String &Right)
    {
        cout << "operator= called" << endl;
        if (&Right != this)
        {
            delete [] Ptr;
            Length = Right.Length;
            Ptr = new char[Length + 1];
            assert(Ptr != 0);
            strcpy(Ptr, Right.Ptr);
        }
        else
            cout << "Attempted assignment of a String
to itself" << endl;
        return *this;
    }

    String &String::operator+=(const String &Right)
    {
        char *TempPtr = Ptr;
        Length += Right.Length;
        Ptr = new char[Length + 1];
        assert(Ptr != 0);
        strcpy(Ptr, TempPtr);
        strcat(Ptr, Right.Ptr);
        delete [] TempPtr;
        return *this;
    }

    int String::operator!() const
    {
        return Length == 0;
    }

    int String::operator==(const String &Right) const
    {
        return strcmp(Ptr, Right.Ptr) == 0;
    }

    int String::operator!=(const String &Right) const
    {
        return strcmp(Ptr, Right.Ptr) != 0;
    }

    int String::operator<(const String &Right) const
    {
        return strcmp(Ptr, Right.Ptr) < 0;
    }

    int String::operator>(const String &Right) const
```

```
{
    return strcmp(Ptr, Right.Ptr) > 0;
}

int String::operator>=(const String &Right) const
{
    return strcmp(Ptr, Right.Ptr) >= 0;
}

int String::operator<=(const String &Right) const
{
    return strcmp(Ptr, Right.Ptr) <= 0;
}

char &String::operator[](int Subscript)
{
    assert(Subscript >= 0 && Subscript < Length);
    return Ptr[Subscript];
}

String &String::operator()(int Index, int SubLength)
{
    assert(Index >= 0 && Index < Length && SubLength
>= 0);
    String *SubPtr = new String;
    assert(SubPtr != 0);
    if ((SubLength == 0) || (Index + SubLength >
Length))
        SubPtr->Length = Length - Index + 1;
    else
        SubPtr->Length = SubLength + 1;
    delete SubPtr->Ptr;
    SubPtr->Ptr = new char[SubPtr->Length];
    assert(SubPtr->Ptr != 0);
    strncpy(SubPtr->Ptr, &Ptr[Index], SubPtr->Length);
    SubPtr->Ptr[SubPtr->Length] = '\0';
    return *SubPtr;
}

int String::GetLength() const
{
    return Length;
}

ostream &operator<<(ostream &Output, const String &S)
{
    Output << S.Ptr;
    return Output;
}

istream &operator>>(istream &Input, String &S)
{
    char Temp[100];
    Input >> setw(100) >> Temp;
    S = Temp;
    return Input;
}

int main()
{
    String S1("happy"), S2(" birthday"), S3;
    cout << "S1 is \"" << S1 << "\"; S2 is \"" << S2
        << "\"; S3 is \"" << S3 << '\"' << endl
        << "The results of comparing S2 and S1:"
```

```

<< endl
        << "S2 == S1 yields " << (S2 == S1) <<
endl
        << "S2 != S1 yields " << (S2 != S1) <<
endl
        << "S2 > S1 yields " << (S2 > S1) <<
endl
        << "S2 < S1 yields " << (S2 < S1) <<
endl
        << "S2 >= S1 yields " << (S2 >= S1) <<
endl
        << "S2 <= S1 yields " << (S2 <= S1) <<
endl;
    cout << "Testing !S3:" << endl;
    if (!S3)
    {
        cout << "S3 is empty; assigning S1 to S3;"
<< endl;
        S3 = S1;
        cout << "S3 is \"" << S3 << "\"" << endl;
    }
    cout << "S1 += S2 yields S1 = ";
    S1 += S2;
    cout << S1 << endl;
    cout << "S1 += \" to you\" yields" << endl;
    S1 += " to you";
    cout << "S1 = " << S1 << endl;
    cout << "The substring of S1 starting at" << endl
        << "location 0 for 14 characters, S1(0,
14), is: "
        << S1(0, 14) << endl;
    cout << "The substring of S1 starting at" << endl
        << "location 15, S1(15, 0), is: "
        << S1(15, 0) << endl; // 0 is "to end of
string"
    String *S4Ptr = new String(S1);
    cout << "*S4Ptr = " << *S4Ptr << endl;
    cout << "assigning *S4Ptr to *S4Ptr" << endl;
    *S4Ptr = *S4Ptr;
    cout << "*S4Ptr = " << *S4Ptr << endl;
    delete S4Ptr;
    S1[0] = 'H';
    S1[6] = 'B';
    cout << "S1 after S1[0] = 'H' and S1[6] = 'B' is:
"<< S1 << endl;
    cout << "Attempt to assign 'd' to S1[30] yields:"
<< endl;
    S1[30] = 'd'; //Loi: Chi so vuot khoi mien!!!
    return 0;
}

```



F1 Help **F2** Save **F3** Open **Alt-F9** Compile **F9** Make **F10** Menu

28. 

kết quả

```

Conversion constructor: happy
Conversion constructor: birthday
Conversion constructor:
S1 is "happy"; S2 is " birthday"; S3 is "
The results of comparing S2 and S1:
S2 == S1 yields 0
S2 != S1 yields 1
S2 > S1 yields 0
S2 < S1 yields 1
S2 >= S1 yields 0
S2 <= S1 yields 1
Testing !S3:
S3 is empty; assigning S1 to S3;
operator= called
S3 is "happy"
S1 += S2 yields S1 = happy birthday
S1 += " to you" yields
Conversion constructor: to you
Destructor: to you
S1 = happy birthday to you
Conversion constructor:
The substring of S1 starting at
location 0 for 14 characters, S1(0, 14), is happy birthday
Conversion constructor:
The substring of S1 starting at
location 15, S1(15, 0), is: to you
Copy constructor: happy birthday to you
*S4Ptr = happy birthday to you
assigning *S4Ptr to *S4Ptr
operator= called
Attempted assignment of a String to itself
*S4Ptr = happy birthday to you
Destructor: happy birthday to you
S1 after S1[0] = 'H' and S1[6] = 'B' is: Happy Birthday to you
Attempt to assign 'd' to S1[30] yields:
Assertion failed: Subscript >= 0 && Subscript < Length, file CT4_18.CPP, line 12
5

```

D/ BÀI TẬP TỰ GIẢI

Câu hỏi trắc nghiệm

Câu 1:

Định nghĩa nào đúng cho toán tử nhập(>>) của một lớp T

- a) istream& operator>>(istream&);
- b) istream& operator>>(istream);
- c) friend istream& operator>>(istream&, T&);
- d) friend istream& operator>>(istream&, T);

Câu 2

Định nghĩa nào phù hợp nhất cho toán tử lấy thành phần([]) của lớp mảng A

- a) `int operator [] (int)`
- b) `int& operator [] (int)`
- c) `friend int operator [] (A&, int)`
- d) `friend int& operator [] (A&, int)`

Câu 3:

Chỉ ra cách định nghĩa toán tử cho lớp T bị sai

- a) `T operator-(T&)`
- b) `T operator-()`
- c) `T operator-()`
- d) `friend T operator +(T&);`
- e) `T operator +(T&);`

Câu 4:

Chỉ ra cách định nghĩa toán tử cho lớp T bị sai

- a) `T& operator ++()`
- b) `T operator ++();`
- c) `T& operator++(int);`
- d) `T&operator++(float)`

Câu 5:

Cho lớp A

```
class A
{
    public:
        operator int ();
};
A a;
int i=a;(1)
float f=a;(2)
```

Chỉ ra dòng nào có lỗi

- a) Chỉ dòng 1 lỗi
- b) Chỉ dòng 2 lỗi
- c) Cả hai dòng đều lỗi
- d) Không dòng nào lỗi

Bài tập**Bài 1:**

Tạo kiểu dữ liệu Date biểu diễn ngày, tháng, năm. Cài đặt các toán tử để tính một ngày trước hoặc sau một ngày xác định nào đó, tính khoảng cách thao ngày giữa hai ngày xác định, tính thứ trong tuần của ngày. Các toán tử vào ra cho ngày.

Bài 2:

Để lưu trữ một ma trận đối xứng thì không cần đủ ô nhớ cho tất cả các phần tử của nó. Xây dựng lớp biểu diễn ma trận đối xứng có các toán tử truy nhập từng phần tử của ma trận. Chỉ sử dụng đủ lượng bộ nhớ cần thiết để lưu ma trận đối xứng

Bài 3

Xây dựng lớp biểu diễn các đa thức với các toán tử cộng, trừ, nhân, chia và đảo dấu. Định nghĩa toán tử xuất để kết xuất.

Bài 4:

Xây dựng một lớp map cho phép biểu diễn một ánh xạ từ một chuỗi kí tự thành một giá trị số nguyên. Định nghĩa toán tử [] cho lớp để có thể sử dụng ánh xạ theo cách như ["abc"]->5

Bài 5

Xây dựng một lớp biểu diễn các vector n chiều với các toán tử cộng, trừ, tích có hướng hai vector và tích vô hướng một vector với một số thực. Định nghĩa toán tử cho phép truy nhập các thành phần của vector.

KỸ THUẬT THỪA KẾ

MỤC TIÊU CỦA BÀI NÀY GIÚP NGƯỜI HỌC

- Cài đặt được sự thừa kế
- Sử dụng các thành phần của lớp cơ sở
- Định nghĩa lại các hàm thành phần
- Truyền thông tin giữa các hàm thiết lập của lớp dẫn xuất và lớp cơ sở
- Các loại dẫn xuất khác nhau và sự thay đổi trạng thái của các thành phần lớp cơ sở.
- Sự tương thích giữa các đối tượng của lớp dẫn xuất và lớp cơ sở
- Toán tử gán và thừa kế
- Hàm ảo và tính đa hình

A/ NHẮC LẠI LÝ THUYẾT

Thừa kế nâng cao khả năng sử dụng lại của các đoạn mã chương trình.

Người lập trình có thể khai báo lớp mới thừa thừa kế dữ liệu và hàm thành phần từ lớp cơ sở đã được định nghĩa trước đó. Ta gọi lớp mới là lớp dẫn xuất.

Trong đơn thừa kế, một lớp chỉ có thể có một lớp cơ sở. Trong đa thừa kế cho phép một lớp là dẫn xuất của nhiều lớp

Lớp dẫn xuất thường bổ sung các thành phần dữ liệu và các hàm thành phần trong định nghĩa, ta nói lớp dẫn xuất cụ thể hơn so với lớp cơ sở và vì vậy thường mô tả một lớp các đối tượng có phạm vi hẹp hơn lớp cơ sở.

Lớp dẫn xuất không có quyền truy nhập đến các thành phần **private** của lớp cơ sở. Tuy nhiên lớp cơ sở có quyền truy xuất đến các thành phần công cộng và được bảo vệ(**protected**).

Hàm thiết lập của lớp dẫn xuất thường tự động gọi các hàm thiết lập của các lớp cơ sở để khởi tạo giá trị cho các thành phần trong lớp cơ sở.

Hàm hủy bỏ được gọi theo thứ tự ngược lại.

Thuộc tính truy nhập **protected** là mức trung gian giữa thuộc tính public và **private**. Chỉ có các hàm thành phần và hàm bạn của lớp cơ sở và lớp dẫn xuất có quyền truy xuất đến các thành phần **protected** của lớp cơ sở.

Có thể định nghĩa lại các thành phần của lớp cơ sở trong lớp dẫn xuất khi thành đó không còn phù hợp trong lớp dẫn xuất.

Có thể gán nội dung đối tượng lớp dẫn xuất cho một đối tượng lớp cơ sở. Một con trỏ lớp dẫn xuất có thể chuyển đổi thành con trỏ lớp cơ sở.

Hàm ảo được khai báo với từ khoá virtual trong lớp cơ sở.

Các lớp dẫn xuất có thể đưa ra các cài đặt lại cho các hàm ảo của lớp cơ sở nếu muốn, trái lại chúng có thể sử dụng định nghĩa đã nêu trong lớp cơ sở.

Nếu hàm ảo được gọi bằng cách tham chiếu qua tên một đối tượng thì tham chiếu đó được xác định dựa trên lớp của đối tượng tương ứng.

Một lớp có hàm ảo không có định nghĩa(hàm ảo thuần túy) được gọi là lớp trừu tượng. Các lớp trừu tượng không thể dùng để khai báo các đối tượng nhưng có thể khai báo con trỏ có kiểu lớp trừu tượng.

B. MỘT SỐ LƯU Ý (Các lỗi thường gặp, một số thói quen lập trình tốt...)

☛ Các lỗi thường gặp

- Cho con trỏ lớp dẫn xuất chỉ đến đối tượng lớp cơ sở mà không đảm bảo chắc chắn rằng phiên bản mới của hàm trong lớp dẫn xuất cũng trả về cùng giá trị như phiên bản cũ của hàm.
- Khai báo đối tượng của lớp trừu tượng

➤ Khai báo hàm thiết lập là hàm ảo.

☛ Một số thói quen lập trình tốt

Khi thừa kế các khả năng không cần thiết trong lớp dẫn xuất, tốt nhất nên định nghĩa lại chúng.

C/ BÀI TẬP MẪU

Ví dụ 1:

Giả sử có các lớp như trong khai báo. Chỉ ra các lỗi sai cho các lệnh của chương trình viết dưới đây.

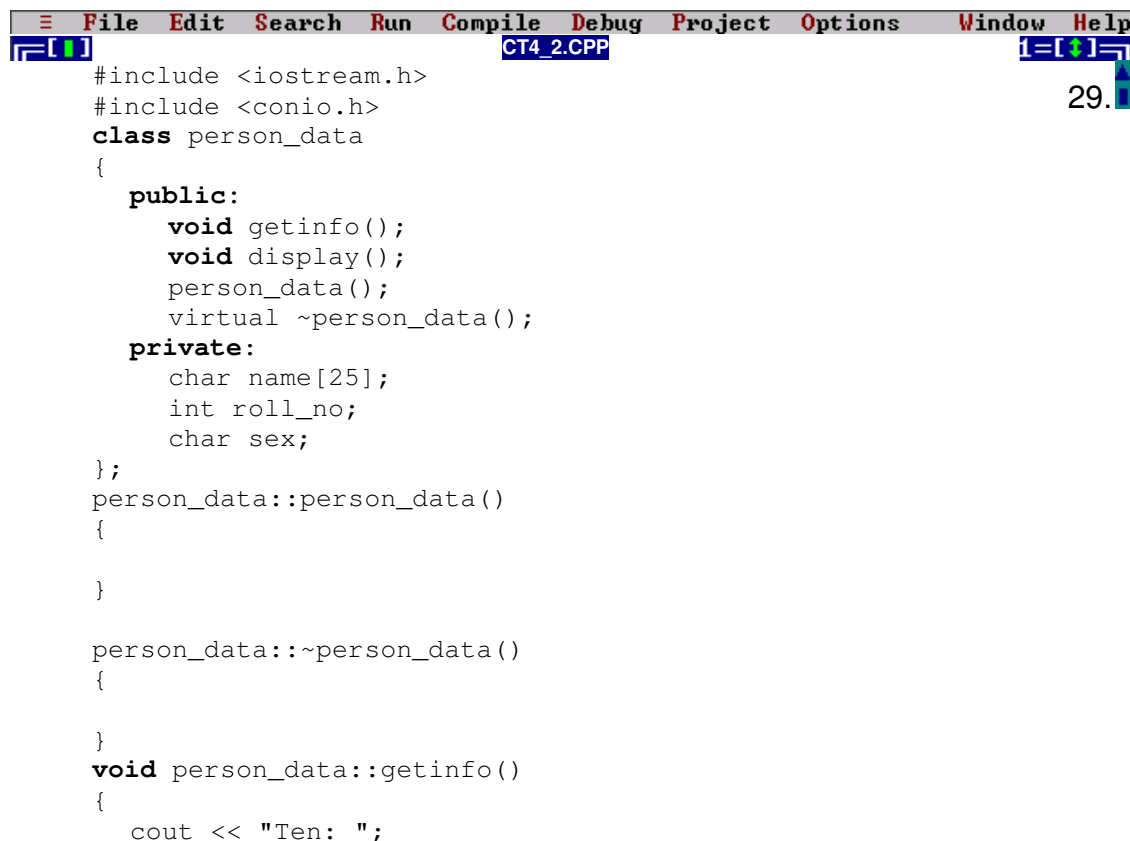
```
class A
{
    public:
        void func();
};
class B: private class A
{
};
A a;
B b;
a.func();
A* pA =&b;
B* pB=&a;
```

Lời giải

Lời gọi b.func() có lỗi bởi vì lớp B kế thừa lớp A theo chế độ **private**. Do vậy, tất cả các thành phần của A sẽ là **private** trong B, hơn nữa không thể truy nhập vào một thành phần **private**.

Một đối tượng của lớp dẫn xuất cũng có thể coi là đối tượng của lớp cơ sở. Do vậy, khai báo A* pA=&b là hoàn toàn đúng. Nhưng điều ngược lại là không đúng, nên khai báo B* pB=&a sẽ gây lỗi khi biên dịch.

Ví dụ 2: Quản lý học viên



```
File Edit Search Run Compile Debug Project Options Window Help
CT4_2.CPP 29.
#include <iostream.h>
#include <conio.h>
class person_data
{
    public:
        void getinfo();
        void display();
        person_data();
        virtual ~person_data();
    private:
        char name[25];
        int roll_no;
        char sex;
};
person_data::person_data()
{

}

person_data::~~person_data()
{

}
void person_data::getinfo()
{
    cout << "Ten: ";
```

```
        cin>> name;
        cout<<"So: ";
        cin>>roll_no;
        cout<<"Gioi tinh(F/M) : ";
        cin>> sex;
    }

    void person_data:: display()
    {
        cout<<name<<"\t";
        cout<<roll_no<<"\t";
        cout<<sex<<"\t";
    }
    //-----
    class academics
    {
    public:
        void getinfo();
        void display();
        academics();
        virtual ~academics();
    private:
        char course_name[25];
        int semester;
        char grade[3];
    };

    academics::academics()
    {

    }

    academics::~~academics()
    {

    }

    void academics::getinfo()
    {
        cout<<"Ten khoa (BA/MBA/MCA etc)? ";
        cin>>course_name;
        cout<< "Hoc ky (1/2/3/...)? ";
        cin>>semester;
        cout<<"muc do (A,B,B+,B-..) ? ";
        cin>>grade;
    }

    void academics::display()
    {
        cout<<course_name<<"\t";
        cout<<semester<<"\t";
        cout<<grade<<"\t";
    }
    //-----

    class stud_scholarship :
        public person_data,
        public academics
    {
    public:
        void getinfo();
        void display();
        stud_scholarship();
        virtual ~stud_scholarship();
```

```

        private:
            float amount;

    };
    stud_scholarship::stud_scholarship()
    {

    }

    stud_scholarship::~stud_scholarship()
    {

    }

    void stud_scholarship::getinfo()
    {
        person_data::getinfo();
        academics::getinfo();
        cout<<"Su ho tro ";
        cin>>amount;
    }
    void stud_scholarship::display()
    {
        person_data::display();
        academics::display();
        cout<<amount<<endl;
    }
    //-----
    int main()
    {
        stud_scholarship obj;
        cout<<"Nhap cac thong tin sau: "<<endl;
        obj.getinfo();
        cout<<endl;
        cout<<"Ten      So  Gioi tinh    Khoa    Hoc ky    Muc do";
        cout<<"      Amount"<<endl;
        obj.display();

        return 0;
    }

```

30. 

F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

D/ BÀI TẬP TỰ GIẢI

Bài 1: Xây dựng lớp Stack với các thao tác cần thiết. Từ đó hãy dẫn xuất từ lớp Stack để đổi một số nguyên dương sang hệ đếm bất kỳ.

Bài 2: Hãy xây dựng các lớp cần thiết trong phân cấp hình 5.2

Bài 3: Hãy xây dựng các lớp cần thiết trong phân cấp hình 5.3 để tính diện tích (hoặc diện tích xung quanh) và thể tích.

Bài 4: Viết một phân cấp kế thừa cho các lớp Quadrilateral (hình tứ giác), Trapezoid (hình thang), Parallelogram (hình bình hành), Rectangle (hình chữ nhật), và Square (hình vuông). Trong đó Quadrilateral là lớp cơ sở của phân cấp.