# BÀI THỰC TẬP 04 – LẬP TRÌNH CÀI ĐẶT ĐƠN KẾ THỪA

## A. MỤC TIÊU

Cung cấp cho sinh viên các kỹ thuật về lập trình hướng đối tượng trong C++:

- Hiểu khái niệm kế thừa

**-** Hiểu và vận dụng các khái niệm về hàm trong lập trình hướng đối tượng: nạp chồng, ghi đè hàm.

**-** Biết cách cài đặt các bài toán vận dụng tính đơn kế thừa trong C++.

## B. NỘI DUNG

* Khai báo định lớp sử dụng đơn kế thừa.
* Sử dụng các toán tử phạm vi.
* Hàm tạo hàm hủy trong kế thừa.
* Các kiểu kế thừa.
* Tạo mảng các đối tượng, giải quyết các bài toán trong mối quan hệ đơn kế thừa.
* Viết chương trình đơn giản sử dụng đơn kế thừa.

## C. YÊU CẦU PHẦN CỨNG, PHẦN MỀM

* Phần cứng: Dung lượng RAM từ 1GB
* Phần mềm: Hệ điều hành Windows, C-free 5.0

## D. KẾT QUẢ SAU KHI HOÀN THÀNH

- Xây dựng được ứng dụng với nhiều lớp được tổ chức theo sự phân cấp kế thừa.

- Cài đặt hàm được hàm tạo, hàm hủy trong kế thừa.

* Sử dụng thành thạo toán tử phạm vi.
* Ghi đè hiệu chỉnh lại nội dung của một phương thức ở lớp dẫn xuất.

## E. TÓM TẮT LÝ THUYẾT

1. **Khái niệm kế thừa**

Kế thừa cho phép các lớp được xây dựng từ các lớp khác đã có. Một lớp được xây dựng kế thừa một lớp khác gọi là lớp dẫn xuất (derived class). Lớp dùng để xây dựng lớp dẫn xuất gọi là lớp cơ sở (base class). Lớp dẫn xuất sẽ kế thừa các thành phần (dữ liệu, hàm) của lớp cơ sở, đồng thời bổ sung các thành phần mới.

Một số trường hợp sau đây không cho phép sử dụng tính kế thừa.

* Hàm tạo.
* Hàm hủy.
* Các toán tử mới do người dùng định nghĩa.
* Các toán tử gán do người dùng định nghĩa
* Quan hệ friend.

1. **Đơn kế thừa**

Trong mối quan hệ kế thừa, lớp chỉ có một lớp cơ sở gọi là đơn kế thừa.

**a. Cú pháp khai báo đơn kế thừa**

|  |
| --- |
| *class <Tên\_lớp\_cơ\_sở>*  *{ ……………… // Thân lớp cơ sở*  *};*  *class <Tên\_lớp\_dẫn\_xuất> : Kiểu\_thừa\_kế <Tên\_lớp\_cơ\_sở>*  *{*  *…………… // Thân lớp dẫn xuất*  *};* |

Trong đó: Kiểu\_thừa\_kế có thể là public, private, protected. Nếu được bỏ trống chương trình ngầm hiểu là private. Tuy nhiên khuyến nghị không nên bỏ trống mục này.

**b. Toán tử phạm vi**

Khi đối tượng lớp dẫn xuất muốn chỉ định thành phần cùng tên trong lớp cơ sở, phải sử dụng tên lớp cơ sở và toán tử phạm vi :: đặt trước tên thành phần đó với cú pháp:

<Tên\_lớp\_cơ\_sở> :: <Tên\_thành\_phần>

**c. Hàm tạo và hàm hủy trong kế thừa.**

Cú pháp định nghĩa hàm tạo lớp dẫn xuất:

|  |
| --- |
| *Hàm\_tạo\_lớp\_dẫn\_xuất(Danh sách tham số):Hàm tạo\_lớp\_cơ\_sở (Danh sách tham số)*  *{*  *……// Thân hàm tạo lớp dẫn xuất*  *}* |

* Nếu lớp cơ sở và lớp dẫn xuất không định nghĩa hàm tạo thì hàm tạo ngầm định của hai lớp sẽ đươc gọi theo đúng thứ tự.
* Nếu lớp cơ sở và lớp dẫn xuất có định nghĩa hàm tạo thì khi định nghĩa hàm tạo lớp dẫn xuất luôn mô tả một lời gọi tới một trong các hàm tạo tương ứng ở lớp cơ sở. Trong đó, hàm tạo không tham số của lớp dẫn xuất có thể có hoặc không cần mô tả lại một lời gọi đến hàm tạo không tham số lớp cơ sở.

Hàm hủy thứ tự gọi ngược lại với hàm tạo. Khi hủy đối tượng hàm hủy lớp dẫn xuất được gọi trước, rồi đến hàm hủy lớp cơ sở.

**d. Phạm vi truy xuất**

* Lớp dẫn xuất chỉ được truy nhập đến thành phần protected và public của lớp cơ sở không được phép truy nhập đến thành phần private.
* Các kiểu kế thừa:

+ Nếu một lớp dẫn xuất kế thừa lớp cơ sở là public thì mọi thành phần public của lớp cơ sở sẽ trở thành thành phần public của lớp dẫn xuất, thành phần protected lớp cơ sở sẽ trở thành thành phần protected của lớp dẫn xuất.

+ Nếu một lớp dẫn xuất kế thừa lớp cơ sở là private thì mọi thành phần public, protected của lớp cơ sở sẽ trở thành thành phần private của lớp dẫn xuất.

+ Nếu một lớp dẫn xuất kế thừa lớp cơ sở là protected thì mọi thành phần public, protected của lớp cơ sở sẽ trở thành thành phần protected của lớp dẫn xuất.

+ Nếu không có từ khoá chỉ định kế thừa từ lớp cơ sở thì lớp dẫn xuất ngầm định là kế thừa private.

|  |  |  |  |
| --- | --- | --- | --- |
| **Thành phần lớp cơ sở** | **Kế thừa public** | **Kế thừa private** | **Kế thừa protected** |
| **private** | \_ | \_ | \_ |
| **public** | public | private | protected |
| **protected** | protected | private | protected |

Bảng 4. 1 Các kiểu kế thừa

## F. HƯỚNG DẪN CHI TIẾT

**Bài 1:** Xây dựng lớp bệnh nhân (BenhNhan) gồm các thuộc tính sau: tên, quê quán, năm sinh. Trong lớp có định nghĩa các hàm thành phần:

- Nhập thông tin một bệnh nhân.

- Xuất thông tin một bệnh nhân.

Xây dựng lớp bệnh án (BenhAn) kế thừa từ lớp bệnh nhân và bổ sung thêm thuộc tính: tên bệnh án, số tiền viện phí và lớp này định nghĩa các hàm thành phần:

- Nhập thông tin một bệnh án.

- Xuất thông tin một bệnh án.

- Tính tuổi hiện tại của bệnh nhân.

Chương trình chính thực hiện các chức năng:

- Nhập danh sách n bệnh án.

- Sắp xếp danh sách bệnh án theo tuổi giảm dần của các bệnh nhân.

- In ra màn hình thông tin bệnh án của những bệnh nhân tuổi>50.

- Hiển thị thông tin bệnh án của bệnh nhân có viện phí cao nhất.

**Hướng dẫn:**

Trong bài thực hành này hướng dẫn sinh viên cài đặt đơn kế thừa, cách khai báo định nghĩa lớp dẫn xuất, sử dụng toán tử phạm vi, ghi đè lại hàm trùng tên trong lớp dẫn xuất.

Dựa và yêu cầu đề bài, cần quản lý hai lớp: BenhNhan và BenhAn, Lớp BenhAn kế thừa từ lớp BenhNhan. Xây dựng các lớp như sơ đồ kế thừa sau:

|  |
| --- |
| **BenhNhan** |
| * hoten: char[] * quequan: char[]   - namsinh: int |
| + nhap(): void  + xuat(): void |

|  |
| --- |
| **BenhAn** |
| * tenba: char[] * tienvp: double |
| + nhap(): void  + xuat(): void  + tuoi(): int |

**Bước 1**: Khai báo định nghĩa lớp BenhNhan.

|  |
| --- |
| **class BenhNhan**  {  char hoten[25];  char quequan[30];  int namsinh;  **public**:  **void nhap()**  {  cout<<"Nhap ten: ";  fflush(stdin); cin.getline(hoten,25);  cout<<"Nhap que quan: "; fflush(stdin); cin.getline(quequan,30);  cout<<"Nhap nam sinh: "; cin>>namsinh;  }  **void xuat()**  {  cout<<"\nTen: "<<hoten;  cout<<"\nQue quan: "<<quequan;  cout<<"\nNam sinh: "<<namsinh;  }  }; |

**Bước 2:** Xây dựng lớp BenhAn kế thừa tứ lớp BenhNhan.

- Khai báo lớp BenhAn kế thừa lớp BenhNhan với kiểu kế thừa public.

|  |
| --- |
| **class BenhAn: public BenhNhan** {  }; |

- Khai báo các thành phần dữ liệu riêng cần bổ sung.

|  |
| --- |
| **private:**  char tenba[30];  double tienvp; |

- Định nghĩa hàm nhap(), xuat() để nhập, xuất thông tin một bệnh án.

|  |
| --- |
| **void nhap()**  {  BenhNhan::nhap();**// Sử dựng toán tử phạm vi gọi đến hàm //trùng tên trong lớp cơ sở**  cout<<"Nhap ten benh an: "; fflush(stdin); gets(tenba);  cout<<"Nhap tien vien phi: "; cin>>tienvp;  cout<<endl;  } |

Hàm nhap() trong lớp BenhAn được định nghĩa trùng tên của lớp cơ sở. Cách viết này được gọi đã ghi đè lên hàm đã có ở lớp cơ sở. Lúc này trong lớp BenhAn có hai hàm nhap(), vì vậy trong thân hàm nhap() của lớp BenhAn muốn gọi lại hàm trùng tên của lớp cơ sở BenhNhan cần dùng toán tử phạm vi “::”. Cách viết là ghi rõ tên lớp kèm toán tử phạm vi trước tên thành phần trùng tên cần gọi:

*BenhNhan::nhap();*

Tương tự như vậy khi định nghĩa hàm xuat().

|  |
| --- |
| **void xuat()**  {  BenhNhan::xuat();  **// Sử dựng toán tử phạm vi gọi đến hàm //trùng tên trong lớp cơ sở**  cout<<"\nTen benh an: "<<tenba;  cout<<"\nTien vien phi: "<<tienvp;  cout<<”\nTuoi: “<<tuoi();  cout<<endl;  } |

**Bước 3:** Xây dựng hàm main() để tạo danh sách bệnh án và thực hiện yêu cầu.

- Khai báo và nhập danh sách BenhAn.

|  |
| --- |
| BenhAn a[50];  int n,i;  do  {  cout<<"Nhap so benh an n= ";  cin>>n;  }  while (n<0);  for (i=0; i<n; i++)  {  cout<<"Nhap thong tin thu "<<i+1<<" \n";  a[i].nhap();  } |

- Xuất danh sách

|  |
| --- |
| cout<<"\n------------\n";  for (i=0; i<n; i++)  {  cout<<"\n-Benh nhan thu : "<<i+1<<" \n";  a[i].xuat();  } |

- Sắp xếp danh sách theo chiều tăng dần của tuổi

Cần bổ sung hàm tính tuổi tuoi() là hàm có quyền public trong lớp BenhAn. Trước khi định nghĩa hàm tính tuổi, trong lớp BenhNhan cần viết thêm hàm GetNs () với chức năng trả về giá trị năm sinh có quyền public. Bởi lớp con BenhAn không được phép truy nhập đến thành phần riêng tư namsinh (có quyền private)của lớp BenhNhan. Hàm GetNs() là giải pháp trong trường hợp này.

Hàm lấy giá trị năm sinh trong lớp BenhNhan

|  |
| --- |
| int GetNs()  {  return namsinh;  } |

Hàm tính tuổi trong lớp BenhAn

|  |
| --- |
| int tuoi()  {  time\_t baygio = time(0);  tm \*ltm = localtime(&baygio);  int tuoi=1900 + ltm->tm\_year – GetNs();  } |

Đoạn chương trình sắp xếp danh sách bệnh nhân theo tuổi.

|  |
| --- |
| for (i=0; i<n-1; i++)  for (int j=i+1; j<n; j++)  {  if (a[j].tuoi()>a[i].tuoi())  {  BenhAn c=a[i];  a[i]=a[j];  a[j]=c;  }  } |

- Tìm bệnh nhân có tuổi > 50

|  |
| --- |
| cout<<"\n------------\n";  cout<<"\nDanh sach benh nhan tren 50 tuoi: ";  for (i=0; i<n; i++)  {  if (a[i].tuoi()>50)  a[i].xuat();  } |

- Hiển thị thông tin bệnh nhân có viện phí cao nhất

|  |
| --- |
| BenhAn vpmax=a[0];  for (i=0; i<n; i++)  {  if (vpmax.Getvp()<a[i].Getvp())  {  vpmax=a[i];  }  }  cout<<"\n------------\n";  cout<<"\nThong tin benh nhan co vien phi cao nhat: ";  vpmax.xuat();  } |

Trong hàm main() khi tìm bệnh nhân có viện phí cao nhất, đối tượng của lớp bệnh án cần truy nhập đến thuộc tính viện phí là thành phần dữ liệu dữ riêng trong lớp BenhAn. Khi viết a[i].tienvp; trình biên dịch sẽ báo lỗi (Lỗi truy nhập đến thành phần dữ liệu riêng của lớp BenhAn). Vì vậy, lớp BenhAn cần bổ sung thêm hàm Getvp() với chức năng trả về giá trị tiền viện phí có quyền public.

|  |
| --- |
| float Getvp()  {  return tienvp;  } |

Khi đó, các đối tượng bệnh nhân sẽ gọi đến hàm này để lấy được thông tin viện phí một cách hợp lệ.

vpmax.Getvp();

a[i].Getvp();

Chương trình cài đặt:

|  |
| --- |
| #include <iostream.h>  #include <stdio.h>  #include <conio.h>  **class BenhNhan**  {  char hoten[25];  char quequan[30];  int namsinh;  **public:**  **void nhap()**  {  cout<<"Nhap ten: "; fflush(stdin); cin.getline(hoten,25);  cout<<"Nhap que quan: "; fflush(stdin); cin.getline(quequan,30);  cout<<"Nhap nam sinh: "; cin>>namsinh;  }  **void xuat()**  {  cout<<"\nTen: "<<hoten;  cout<<"\nQue quan: "<<quequan;  cout<<"\nNam sinh: "<<namsinh;  }  **int GetNs()**  {  return namsinh;  }  };  **class BenhAn: public BenhNhan**  {  **private:**  char tenba[30];  double tienvp;  p**ublic:**  **void nhap()**  {  BenhNhan::nhap();  cout<<"Nhap ten benh an: "; fflush(stdin); gets(tenba);  cout<<"Nhap tien vien phi: "; cin>>tienvp;  cout<<”\nTuoi: “<<tuoi();  cout<<endl;  }  **void xuat()**  {  BenhNhan::xuat();  cout<<"\nTen benh an: "<<tenba;  cout<<"\nTien vien phi: "<<tienvp;  cout<<endl;  }  **int tuoi()**  {  time\_t baygio = time(0);  tm \*ltm = localtime(&baygio);  int tuoi=1900 + ltm->tm\_year -GetNs();  return tuoi;  }  **float Getvp()**  {  return tienvp;  }  };  **main()**  {  BenhAn a[50];  int n,i;  do  {  cout<<"Nhap so benh an n= ";  cin>>n;  }  while (n<0);  for (i=0; i<n; i++)  {  cout<<"Nhap thong tin thu "<<i+1<<" \n";  a[i].nhap();  }  cout<<"\n------------\n";  for (i=0; i<n; i++) {  cout<<"\n-Benh nhan thu : "<<i+1<<" \n";  a[i].xuat();  }  for (i=0; i<n-1; i++)  for (int j=i+1; j<n; j++)  {  if (a[j].tuoi()>a[i].tuoi())  {  BenhAn c=a[i];  a[i]=a[j];  a[j]=c;  }  }  cout<<"\n------------\n";  cout<<"\nThong tin benh an giam dan ve tuoi: \n";  for (i=0; i<n; i++){  a[i].xuat();  }  cout<<"\n------------\n";  cout<<"\nDanh sach benh nhan tren 50 tuoi: ";  for (i=0; i<n; i++)  {  if (a[i].tuoi()>50)  a[i].xuat();  }  BenhAn vpmax=a[0];  for (i=0; i<n; i++)  if (vpmax.Getvp()<a[i].Getvp())  {  vpmax=a[i];  }  cout<<"\n------------\n";  cout<<"\nThong tin benh nhan co vien phi cao nhat: ";  vpmax.xuat();  } |

**Mở rộng:** Cài đặt chương trình sử dụng kiểu thừa kế dạng protected, private.

**Bài 2:** Xây dựng lớp mydate gồm các thuộc tính: d (ngày sinh), m (tháng sinh), y (năm sinh). Trong lớp các định nghĩa các hàm thành phần: tạo không tham số, tạo 3 tham số, hủy hiển thị thông tin một đối tượng mydate.

Xây dựng lớp person kế thừa từ lớp mydate và bổ sung thêm thuộc tính: name, address, phone. Lớp này định nghĩa các hàm: Tạo không tham số, tạo 4 tham số, hủy, hiển thị thông tin. Định nghĩa toán tử > so sánh năm sinh của hai person.

Viết chương trình nhập và xuất danh sách n person (n nguyên dương nhập từ bàn phím). Sắp xếp và in lại danh sách person theo chiều giảm tăng dần của năm sinh.

**Hướng dẫn:**

Trong bài thực hành này cung cấp cho sinh viên kiến thức về cách viết hàm tạo, hủy trong lớp dẫn xuất. Cách sử dụng hàm tạo để khởi tạo đối tượng của lớp.

Chú ý: Một lớp có thể định nghĩa nhiều hàm tạo.

Xây dựng các lớp theo sơ đồ kế thừa sau:

|  |
| --- |
| **mydate** |
| * d:int * m: int   - y: int |
| + mydate(d,m,y)  + display(): void |

|  |
| --- |
| **person** |
| * name: char[] * address: char[] * phone: long int |
| + person(d,m,y,name,address, phone): person  + display(): void  + operator > () : bool |

**Bước 1*:*** Tạo lớp cơ sở mydate có hàm tạo không tham số và hàm tạo ba tham số.

|  |
| --- |
| **class mydate**  {  **private:**  int d,m,y;  **public:**  **mydate()**  {  d=0; m=0; y=0;  }  **mydate(int dd,int mm,int yy)**  {  d=dd;m=mm;y=yy;  }  **~mydate()**{}  **void display()**  {  cout<<d<<"-"<<m<<"-"<<y;  }  **int GetNam()**  {  return y;  }  };//hetlop |

**Bước 2:** Định nghĩa lớp person kế thừa từ lớp mydate. Kiểu thừa kế là public.

- Khi định nghĩa hàm tạo không tham số của lớp person không nhất thiết gọi đến hàm tạo không tham số của lớp mydate tuy nhiên hàm tạo có tham số của lớp person phải chỉ rõ gọi đến hàm tạo phù hợp ở lớp mydate.

|  |
| --- |
| **class person:public mydate**  {  **private:**  char name[30];  char address[40];  long int phone;  **public:**  **person():mydate()**  {  strcpy(name,"");  strcpy(address,"");  phone=0;  }  // Ham tao lop dan xuat  **person(char \*ten,int dd,int mm,int yy,char \*dc,long int dt):mydate(dd,mm,yy)**  {  strcpy(name,ten);  strcpy(address,dc);  phone=dt;  }  **~person()**{}  **bool operator > (person x)**  {  return GetNam()>x.GetNam();  }  **void display()**  {  cout<<"\nName :"<<name;  cout<<"\nAddress: "<<address;  cout<<"\nPhone: "<<phone;  cout<<"\nBirthday: "; mydate::display();  cout<<"\n";  }  };//het lop |

**Bước 3:** Viết hàm main() thực hiện các yêu cầu đề bài.

- Khai báo và nhập danh sách các đối tượng.

|  |
| --- |
| **main()**  {  char ten[30],dc[40]; int d,m,y, n; long int dt;  person p[20];  do  {  cout<<"\n Nhap so nguoi n = "; cin>>n;  }  while(n<0);  for(int i=0;i<n;i++)  {  cout<<"\nNhap thong tin nguoi thu "<<i+1<<" : ";  cout<<"\nName: "; cin.ignore(1); gets(ten);  cout<<"Address: ";gets(dc);  cout<<"Phone: ";cin>>dt;  cout<<"Birthday( ngay, thang, nam): ";cin>>d>>m>>y;  p[i]=person(ten,d,m,y,dc,dt);  }  cout<<"\n Hien thi danh sach sau khi nhap :"<<endl;  for(int i=0;i<n;i++) p[i].display(); |

Khai báo danh sách các đối tượng person p[20]; đã sử dụng hàm tạo không tham số để gắn cho mỗi đối tượng được khai báo trong danh sách p.

Lệnh p[i]=person(ten,d,m,y,dc,dt) sử dụng hàm tạo có tham số để khởi tạo thành phần dữ liệu cho các đối tượng p[i].

- Sắp xếp và hiển thị danh sách đã sắp.

|  |
| --- |
| for(int i=0;i<n;i++)  for(int j=i+1;j<n;j++)  if(p[i]>p[j]) // su dung toan tu da viet  {  person tg= p[i];  p[i]=p[j];  p[j]= tg;  }  cout<<"\n Hien thi DS sau sap xep theo nam sinh"<<endl;  for(int i=0;i<n;i++)  p[i].display();  } |

**Mở rộng:** Yêu cầu viết chương trình chỉ sử dụng một hàm tạo để khởi tạo các đối tượng thay vì hai hàm tạo như bài thực hành trên.

**Bài 3:** Xây dựng lớp Mặt hàng gồm:

+ Thuộc tính: mã hàng, tên hàng, đơn giá

+ Phương thức: Tạo không tham số, tạo có 3 tham số, hủy,

+ Định nghĩa toán tử nhập >>

Xây dựng lớp Bánh kẹo kế thừa từ lớp Mặt hàng và bổ sung thêm:

+ Thuộc tính: nơi sản xuất

+ Phương thức: Tạo không tham số, tạo 4 tham số, tính chiết khấu (biết chiết khấu = 1% đơn giá).

+ Định nghĩa toán tử nhập >>, toán tử xuất << .

Chương trình chính thực hiện:

+ Nhập vào danh sách n đối tượng bánh kẹo.

+ In ra màn hình danh sách các mặt hàng có mã khác “MO1” (Sử dụng toán tử)

+ Tìm mặt hàng có mã nhập từ bàn phím. (Sử dụng toán tử)

+ Đếm số lượng mặt hàng có tên là “Banh ngot” và sản xuất ở “Bac Ninh”

+ Sắp xếp danh sách mặt theo chiều giảm dần của đơn giá. (Sử dụng toán tử)

**Hướng dẫn:** Trong bài thực hành này sinh viên chú ý đến cách sử dụng thành phần protected của lớp cơ sở trong lớp dẫn xuất.

Xây dựng các lớp như sơ đồ kế thừa sau:

|  |
| --- |
| **MatHang** |
| # mahang: char[]  # tenhang: char[]  # dongia: float |
| + MatHang()  +MatHang(mahang,tenhang, dongia)  + friend operator >> |

|  |
| --- |
| **BanhKeo** |
| * noisanxuat: char[] |
| + BanhKeo()  +BanhKeo(mahang, tenhang, dongia,noisanxuat)  + chietkhau(): float  + friend operator >>  + friend operator << |

**Bước 1:** Viết lớp MatHang thiết lập các thành phần thuộc tính có quyền protected.

|  |
| --- |
| class MatHang  {  protected:  char mahang[30];  char tenhang[30];  float dongia; |

- Định nghĩa toán tử nhập >>

|  |
| --- |
| friend istream& operator >>(istream &is, MatHang &p)  {  cout<<"nhap vao ma hang :";  cin.ignore(1);  is.getline(p.mahang,30);  cout<<"Nhap vao ten hang ; ";  is.getline(p.tenhang,30);  cout<<"Nhap vao don gia :";  is>>p.dongia;  return is;  } |

**Bước 2:** Định nghĩa lớp MatHang kế thừa từ lớp BanhKeo

|  |
| --- |
| class BanhKeo: public MatHang  {  private:  char noisanxuat[50];  public:  BanhKeo()  {  strcpy(this->noisanxuat," ");  }  BanhKeo(char \*mahang,char \*tenhang,float dongia,char \*noisanxuat) : MatHang(mahang,tenhang,dongia)  {  strcpy(this->noisanxuat,noisanxuat);  }  ~BanhKeo()  {  }  float chietkhau()  {  return (dongia\*10)/100;  }  friend ostream& operator <<(ostream &os, BanhKeo &p)  {  os<<p.mahang<<setw(12)<<p.tenhang<<setw(12)<<p.dongia<<setw(12)<<p.noisanxuat<<p.chietkhau()<<endl;  }  friend istream& operator >>(istream &is, BanhKeo &p)  {  is>>(MatHang&)p;  cout<<"Nhap vao noi san xuat :";  is.getline(p.noisanxuat,50);  }  };// het lop |

Khi định nghĩa hàm toán tử xuất trong lớp BanhKeo các thành phần dữ liệu mahang, tenhang, dongia của lớp MatHang được phép truy nhập tại lớp BanhKeo do những thành phần này có quyền protected.

**Bước 3:** Viết hàm main() hoàn thiện chương trình.

- Khai báo, nhập dữ liệu tương tự bài thực hành 1.

- Tìm danh sách mặt hàng khác mã “MO1”.

Cần bổ sung thêm hàm Getmahang() có quyển public tại lớp BanhKeo bởi thành phần dữ liệu mahang có quyền protected chỉ được phép truy nhập bên ngoài lớp MatHang và trong lớp dẫn xuất của nó là lớp BanhKeo.

|  |
| --- |
| cout<<"\n Cac loai banh keo co ma khac M01 :"<<endl;  for(int i=0;i<n;i++)  if(strcmpi(s[i].Getmahang(),"M01")!=0)  cout<<s[i]<<endl; |

*Sinh viên tự hoàn thiện chương trình.*

**Mở rộng:**

- Yêu cầu xử lý chương trình không sử dụng hàm Getmahang()

- Cài đặt chương trình sử dụng kiểu thừa kế dạng protected, private.

**Bài** **4:** Xây dựng lớp Ấn phẩm gồm:

- Thuộc tính: tên, giá thành.

- Hàm nhập, hàm xuất thông tin một ấn phẩm

Xây dựng lớp Sách kế thừa từ lớp Ấn phẩm và bổ sung thêm:

- Thuộc tính: Số trang.

- Hàm nhập, hàm xuất thông tin một quyển sách

Xây dựng lớp Đĩa CD kế thừa từ lớp Ấn phẩm và bổ sung thêm:

- Thuộc tính: Số phút.

- Hàm nhập, hàm xuất thông tin một đĩa CD

Chương trình chính thực hiện:

- Khai báo danh sách n quyển sách (n nguyên dương nhập từ bàn phím), nhập danh sách và hiển thị những quyển sách có số trang > 500.

- Khai báo danh sách n đĩa CD (n nguyên dương nhập từ bàn phím), nhập danh sách và hiển thị đĩa CD được tìm theo tên (tên cần tìm nhập từ bàn phím).

**Hướng dẫn:**

Bài thực hành này cung cấp cho sinh viên kiến thức về cách viết các lớp dẫn xuất từ lớp cơ sở trong mối quan hệ thừa kế. Một lớp cơ sở có thể cho nhiều lớp dẫn xuất thừa kế.

Xây dựng các lớp như sơ đồ kế thừa sau:

|  |
| --- |
| **AnPham** |
| * ten: char[] * gia: double |
| + nhap(): void  + hienthi(): void |

|  |
| --- |
| **Dia** |
| * sophut: int |
| + Sach::nhap(): void  + Sach::hienthi(): void |

|  |
| --- |
| **Sach** |
| * sotrang: int |
| + Sach::nhap(): void  + Sach::hienthi():void |

**Bước 1:** Xây dựng lớp ấn phẩm Ấn phẩm

|  |
| --- |
| #include <iostream.h>  class AnPham {  protected:  char ten[30];  double gia;  public:  void nhap()  { cout<<"\n Nhap ten: "; cin.ignore(1); gets(ten);  cout<<"\n Nhap gia thanh: "; cin>>gia;  }  void hienthi()  {  cout<<"\nTen :" <<ten;  cout<<"\nGia thanh :"<<gia<<endl;  }  }; |

**Bước 2:** Xây dựng lớp Sach kế thừa từ lớp AnPham

|  |
| --- |
| class sach: public AnPham{  private:  int sotrang;  public:  void nhap()  {  AnPham::nhap();  cout<<"\n Nhap so trang cua sach nay: ";  cin>>sotrang;  }  void hienthi()  {  AnPham::hienthi();  cout<<"So trang :"<<sotrang<<endl;  cout<<endl;  }  friend void KiemTra(sach s[],int n){  for(int i=0;i<n;i++)  if(s[i].sotrang>500)  s[i].hienthi();  }  }; |

**Bước 3:** Xây dựng lớp Dia kế thừa từ lớp AnPham

|  |
| --- |
| class dia: public AnPham {  private:  int sophut;  public:  void nhap()  {  AnPham::nhap();  cout<<"\n Nhap so phut cho dia nay: ";  cin>>sophut;  }  void hienthi()  {  AnPham::hienthi();  cout<<"So phut :"<<sophut<<endl;  cout<<endl;  }  int TimTen(char x[]){  if(strcmp(ten,x)==0)  return 1;  return 0;  }  }; |

**Bước 4: Viết hàm main() thực hiện các chức năng.**

|  |
| --- |
| sach s[50];  cout<<"\n--------Sach co so trang > 500 ------"<<endl;  KiemTra(s,n);  cout<<"\n\nSo dia can xuat ban: "<<endl;  cin>>n;  dia d[50];  for(i=0;i<n;++i)  {  cout<<"\ Dia: "<<i<<endl;  d[i].nhap();  }  char t[30];  cin.ignore(1);  cout<<"\n-Nhap ten dia CD can tim1: ";  cin.getline(t,30);  for(i=0;i<n;i++)  if (d[i].TimTen(t))  d[i].hienthi();  } |

**Bài 5:** Xây dựng lớp person gồm:

- Thuộc tính: name, address, phone;

- Phương thức: Tạo không tham số, tạo có tham số.

Xây dựng lớp officer kế thừa từ lớp person và bổ sung thêm:

- Thuộc tính: salary

- Phương thức: Tạo không tham số, tạo có tham số.

Xây dựng lớp manager kế thừa từ lớp officer và bổ sung thêm:

- Thuộc tính: extra

- Phương thức: Tạo không tham số, tạo có tham số, hiển thị thông tin một đối tượng manager.

Chương trình chính thực hiện khai báo danh sách n đối tượng manager, hiển thị thông tin. Tìm những đối tượng ở “Ha Noi” và có salary khác 10 triệu.

**Hướng dẫn:**

Bài thực hành này cung cấp cho sinh viên kiến thức về cách viết các lớp dẫn xuất từ lớp cơ sở trong mối quan hệ thừa kế, các kiểu kế thừa.

Xây dựng các lớp theo sơ đồ kế thừa như sau: Một lớp dẫn xuất kế thừa từ lớp cơ sở, lớp này lại là lớp cơ sở của lớp dẫn xuất khác.

|  |
| --- |
| **person** |
| * name: char[] * address: char[] * phone: long int |
| + person()  + person(name,addres, phone) |

|  |
| --- |
| **officer** |
| * salary: float |
| + officer()  + officer (name,addres, phone, salary) |

|  |
| --- |
| **manager** |
| * extra: float |
| + manager()  + manager(name,addres, phone, salary, extra) |

**Bước 1:** Xây dựng lớp person

|  |
| --- |
| include<iostream.h>  **class person**  {  **protected:**  char name[30];  char address[40];  long int phone;  **public:**  **person()**{}  **person(char \*ten,char \*dc,long int dt)**  {  strcpy(name,ten);  strcpy(address,dc);  phone=dt;  }  };//het lop |

**Bước 2:** Xây dựng lớp officer kế thừa từ lớp person với kiểu thừa kế private

|  |
| --- |
| **class officer: private person**  {  **protected:**  float salary;  **public:**  **officer()**  {  }  **officer(char \*ten,char \*dc,long int dt,float sal):**  **person(ten,dc,dt**)  {  salary=sal;  }  **bool operator !=( float a )**  {  return salary != a ;  }  **char \* GetAdd()**  {  return address;  }  };//het lop |

**Bước 3:** Xây dựng lớp manager kế thừa từ lớp officer với kiểu thừa kế public.

Sử dụng hàm GetAdd() để lấy ra thành phần dữ địa chỉ address của đối tượng. Hàm này được định nghĩa trong vùng public của lớp officer. Lớp officer kế thừa từ lớp person với kiểu thừa kế là private do đó các thành phần nằm trong vùng protected và public của lớp person sẽ trở thành thành phần có quyền private trong lớp officer (trong đó có thành phần address). Bởi vậy khi lớp manager kế thừa từ lớp officer (mặc dù kiểu kế thừa là public) thì thành phần địa chỉ address vẫn không được truy nhập từ lớp manager bởi vậy hàm GetAdd() là giải pháp phù hợp.

|  |
| --- |
| **class manager: public officer**  {  **private:**  float extra;  **public:**  **manager ()**  {  }  **manager(char \*ten,char \*dc,long int dt,float sal,float pc) :officer(ten,dc,dt,sal)**  {  extra=pc;  }  **void outscreen()**  {  cout<<"\n=============================\n";  cout<<"\nName: "<<Getname();  cout<<"\nAddress: "<<GetAdd();  cout<<"\nPhone: "<<GetPhone();  cout<<"\nSalary: "<<salary;  cout<<"\nExtra: "<<extra;  }  **bool Kt()**  {  if(strcmpi(GetAdd(), "Ha Noi") == 0)  return 1;  return 0;  }  };//het lop |
|  |

**Bước 4:** Cài đặt đoạn tìm kiếm phần tử trong hàm main()

|  |
| --- |
| cout<<"\n Danh sach doi tuong can tim"<<"\n";  for(int i=0;i<5;i++)  {  if((p[i].kt()==1 && p[i]!= 10000000)  p[i].outscreen();  }  } |

*Chương trình hoàn chỉnh sinh viên tự thực hiện.*

**Bài tập tự giải**

**Bài 6:** Xây dựng lớp PET với những thuộc tính: Tuổi, cân nặng. Xây dựng lớp Dog kế thừatừ lớp PET, bổ sung thêm thành phần: màu mắt, sở thích.

- Nhập và xuất danh sách Dog (Sử dụng toán tử).

- Sắp xếp dach sách Dog theo chiều tăng dần của cân nặng (Sử dụng toán tử).

- Tìm Dog có tuổi từ 2 đến 5

- Tính tổng cân nặng của có con Dog.

**Bài 7:** Xây dựng lớp Hàng hóa (HangHoa) gồm các thuộc tính: Mã, tên hàng, đơn giá. Lớp bổ sung thêm các hàm thành phần:

- Hàm tạo, hàm hủy.

- Hàm nhập, hàm xuất.

Xây dựng lớp máy tính (MT) kế thừa từ lớp hàng hóa và bổ sung thêm các thuộc tính: tốc độ, số lượng. Lớp này bổ sung thêm các hàm thành phần:

- Hàm tạo, hàm hủy

- Định nghĩa toán tử nhâp >>, xuất <<

- Hàm tính thành tiền. Biết thành tiền = số lượng \* đơn giá

Xây dựng lớp máy tính xách tay (MTXT) kế thừa từ lớp máy tính và bổ sung thêm các thuộc tính: trọng lượng. Lớp bổ sung thêm các hàm thành phần:

- Hàm tạo, hàm hủy.

- Hàm tính phí bảo trì. Biết phí bảo trì = 5%\*đơn giá.

- Định nghĩa toán tử nhâp >>, xuất <<.

Hoàn thiện chương trình, xây dựng menu thực hiện các chức năng sau:

- Nhập danh sách máy tính xách tay, hiển thị danh sách

- Tìm thông tin máy tính xách tay theo mã máy nhập vào từ bàn phím.

- Liệt kê máy tính xách tay theo tên.

- Tính tổng phí bảo trì của các máy tính.

- Tìm máy tính xách tay có tốc độ nhỏ nhất.

- Sửa thông tin một máy tính xách tay.

- Xóa một máy tính xách tay theo mã cho trước.

- Chèn thêm một máy tính xách tay tại vị trí k. (k nhập từ bàn phím)

- Sắp xếp danh sách theo chiều tăng dần của mã máy.