**BÀI THỰC TẬP 03: TÁI ĐỊNH NGHĨA TOÁN TỬ**

**A. MỤC TIÊU**

Trang bị cho sinh viên kỹ năng định nghĩa và sử dụng toán tử khi viết chương trình, giúp chương trình gọn gàng và sáng sủa hơn. Đồng thời củng cố các kỹ năng viết chương trình hướng đối tượng với C++.

1. **NỘI DUNG**

* Cách định nghĩa các toán tử.
* Sử dụng các toán tử cơ bản để viết chương trình.

**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**

Sinh viên thành thạo cách định nghĩa và sử dụng các toán tử để viết các chương trình hướng đối tượng bằng ngôn ngữ lập trình C++.

**E. TÓM TẮT LÝ THUYẾT**

1. **Khai báo toán tử:**

|  |
| --- |
| Cách viết tên toán tử:  **operator <tên toán tử>**  Cách viết hàm toán tử:  <kiểu\_trả\_về> **operator <tên toán tử> (** [ds\_tham\_số] **)** |

1. **Tái định nghĩa toán tử nhập, xuất**

|  |
| --- |
| Toán tử nhập:  **friend istream& operator>> ( istream &**TênN **,** TênLớp **&**Tênđốitượng **)**  **{**  **…**  **TênN >>…**  **return TênN;**  **}**  Toán tử xuất:  **friend ostream& operator<< ( ostream &**TênX **,** TênLớpTênđốitượng **)**  **{**  **…**  **TênX <<…**  **return TênX;**  **}** |

**3. Tái định nghĩa toán tử cho phép toán**

|  |
| --- |
| Khai báo và định nghĩa các toán tử:  <kiểu\_trả\_về> **operator <tên toán tử> (**[ds\_tham\_số]**)**  Số tham số:   * Nếu hàm toán tử là hàm tự do hoặc hàm bạn của lớp: số tham số phải đúng bằng số toán hạng của phép toán. * Hàm toán tử là hàm thành phần: Có một tham số ngầm định là đối tượng gọi hàm nên chỉ có một tham số tường minh. |

**4. Viết chương trình hoàn chỉnh tái định nghĩa toán tử**

**F. HƯỚNG DẪN CHI TIẾT**

**Bài 1:**

Xây dựng lớp số phức có 2 thành phần thuộc tính phần thực, phần ảo. Viết chương trình nhập 2 số phức và in ra màn hình 2 số phức đó (Yêu cầu nhập xuất bằng toán tử nhập >>, xuất <<)

**Hướng dẫn:**

Toán tử nhập phải định nghĩa như hàm bạn của lớp, nó là hàm tự do được định nghĩa như sau:

|  |
| --- |
| **istream& operator>>(istream &is,SP &a)**  {  cout<<"Nhap phan thuc: ";  **is>>a.pt;**  cout<<"Nhap phan ao: ";  **is>>a.pa;**  return **is**;  } |

Toán tử xuất phải định nghĩa như hàm bạn của lớp, nó là hàm tự do được định nghĩa như sau:

|  |
| --- |
| **ostream& operator<<(ostream &os,SP a)**  **{**  if (a.pa>=0)  **os<<a.pt<<**"+j\*"**<<a.pa<<**endl**;**  else  **os<<a.pt<<"-**j\***"<<fabs(a.pa)<<**endl**;**  return **os;**  **}** |

Trong lớp phải khai báo hàm toán tử nhập >>, xuất << là hàm bạn

|  |
| --- |
| **friend** istream& operator>>(istream &is,SP &a);  **friend** ostream& operator<<(ostream &os,SP a); |

Khi đó, trong chương trình có thể sử dụng nhập, xuất số phức A như sau:

|  |
| --- |
| SP A;  cout<<"Nhap so phuc A: ";  cin>>A;  cout<<"In ra so phuc A: "<< A; |

Hoàn thiện chương trình này như sau:

|  |
| --- |
| #include<iostream.h>  #include<math.h>  class SP  {  float pt,pa;  public:  **friend istream& operator>>(istream &is,SP &a);**  **friend ostream& operator<<(ostream &os,SP a);**  };  **istream& operator>>(istream &is,SP &a)**  {  cout<<"Nhap phan thuc: ";  **is>>a.pt;**  cout<<"Nhap phan ao: ";  **is>>a.pa;**  return **is**;  }  **ostream& operator<<(ostream &os,SP a)**  {  if (a.pa>=0)  **os<<a.pt<<"+j\*"<<a.pa<<endl;**  else  **os<<a.pt<<"-j\*"<<fabs(a.pa)<<endl;**  return **os;**  }  int main()  {  SP A, B;  cout<<"Nhap so phuc A: ";  **cin>>A;**  cout<<"Nhap so phuc B: ";  **cin>>B;**  cout<<"\nIn 2 so phuc: "<<endl;  **cout**<<"A = " **<< A**;  **cout**<<"B = " **<< B**;  return 0;  } |

**Bài 2:**

Xây dựng lớp đa thức bậc 3 DaThuc3 gồm các thuộc tính hệ số bậc 3, bậc 2, bậc 1 và bậc 0 lần lượt là a, b, c, d (a, b, c, d nguyên) là các thuộc tính riêng. Trong lớp có định nghĩa các hàm sau:

* Hàm khởi tạo với các tham số có giá trị ngầm định bằng 0
* Hàm toán tử nhập >> đi với cin để nhập dữ liệu cho 1 đối tượng đa thức
* Hàm toán tử xuất << đi với cout để in đa thức ra màn hình ở dạng (ax3 + bx2 + cx + d )
* Hàm thành phần tính tổng, hiệu 2 đa thức.

Viết chương trình nhập 2 đa thức và in ra màn hình tổng, hiệu 2 đa thức đó[1].

**Hướng dẫn:**

Xây dựng lớp như các chương trình trước, lưu ý xây dựng hàm toán tử nhập, xuất 1 đa thức như sau:

* Hàm toán tử nhập >> đi với cin, xuất << đi với cout phải là hàm bạn của lớp đa thức
* Định nghĩa hàm toán tử nhập, xuất như hàm tự do:
* Trong chương trình chính có thể sử dụng toán tử >> đi với cin, xuất << đi với cout để nhập, xuất một đối tượng đa thức như đối tượng thông thường.

**Bài 3:**

Xây dựng lớp giáo viên GV gồm các thuộc tính họ tên ht (kiểu char[30]), tuổi t (kiểu số nguyên), bằng cấp cao nhất được đào tạo bc (kiểu char[15]), chuyên ngành (kiểu char[20]), bậc lương bl (kiểu float) là các thuộc tính riêng. Trong lớp có định nghĩa các hàm công cộng sau:

* Hàm toán tử nhập >> đi với cin để nhập dữ liệu cho các thông tin trên của 1 giáo viên.
* Hàm toán tử xuất << đi với cout để xuất các thông tin trên của giáo viên ra màn hình
* Hàm tính tiền lương cơ bản lcb biết rằng lcb = bl \*610

Viết chương trình nhập vào n(n nguyên dương nhập từ bàn phím) giáo viên. In ra màn hình danh sách các giáo viên có tiền lương nhỏ hơn 2000.[2]

Với 1 giáo viên có họ tên được nhập vào từ bàn phím cho biết trong danh sách có giáo viên đó không? Nếu có đưa ra lương của giáo viên đó. Nếu không có chèn thông tin giáo viên đó vào cuối danh sách.

**Hướng dẫn:**

Định nghĩa hàm toán tử nhập >> đi với cin:

|  |
| --- |
| **istream& operator>>(istream &is,GV &gv)**  **{**  fflush(stdin);  cout<<"\nnhap ho ten: ";  **is.getline(gv.ht,30);**  cout<<"\nNhap tuoi : ";  **is>>gv.tuoi;**  fflush(stdin);  cout<<"\nbang cap: ";  **is.getline(gv.bc,15);**  fflush(stdin);  cout<<"\nchuyen nganh ";  **is.getline(gv.cn,20);**  cout<<"\nNhap bac luong: ";  **is>>gv.bl;**  **return** **is**;  **}** |

Định nghĩa hàm toán tử xuất << đi với cout: (trong hàm này có gọi đến phương thức luong() của lớp)

|  |
| --- |
| **ostream& operator<<(ostream& os,GV gv)**  **{**  **os<<gv.ht**<<setw(12)<<**gv.tuoi**<<setw(12)  <<**gv.bc**<<setw(12)<<**gv.cn**<<setw(12)<<**gv.luong();**  **return os;**  **}** |

Trong lớp phải khai báo hàm toán tử nhập >>, xuất << là hàm bạn:

|  |
| --- |
| **friend** istream& **operator>>**(istream &is,GV &gv);  **friend** ostream& **operator<<**(ostream& os,GV gv); |

Trong chương trình chính, khai báo mảng các đối tượng A có n phần tử, n nhập từ bàn phím:

|  |
| --- |
| **GV A[50];**  int n;  cout<<"Nhap so giao vien: ";  **cin>>n;** |

Nhập thông tin cho n giáo viên, sử dụng vòng lặp for nhập cho các đối tượng từ A[0] đến A[n-1] với toán tử nhập >> đi với cin như sau:

|  |
| --- |
| **for(i = 0;i < n; i++)**  **{**  cout<<"\nNhap thong tin giao vien thu "<<i+1<<endl;  **cin>>A[i];**  **}** |

In thông tin giáo viên có tiền lương nhỏ hơn 2000, sử dụng vòng lặp for duyệt lần lượt các đối tượng từ A[0] đến A[n-1], nếu đối tượng nào có luong() nhỏ hơn 2000 thì in đối tượng đó ra màn hình bằng toán tử << đi với cout như sau:

|  |
| --- |
| **for(i = 0;i < n; i++)**  **{**  if(A[i].luong()<2000)  **cout<<A[i];**  **}** |

Với 1 giáo viên có họ tên được nhập vào từ bàn phím cho biết trong danh sách có giáo viên đó không? Nếu có đưa ra lương của giáo viên đó. Nếu không có chèn thông tin giáo viên đó vào cuối danh sách. Để thực hiện yêu cầu này, thực hiện lần lượt các bước sau:

* Khai báo 1 đối tượng gvtk và nhập dữ liệu cho đối tượng này bằng lệnh cin>>gvtk;
* Kiểm tra trong danh sách có giáo viên đó không: chúng ta duyệt lần lượt các đối tượng từ đầu đến cuối
* Nếu có đối tượng có họ tên trùng với họ tên cần tìm kiếm thì đánh dấu có tìm thấy và in thông tin về lương của đối tượng đó ra màn hình.
* Ngược lại hết danh sách mà không tìm thấy thì thông báo không có giáo viên đó trong danh sách và chèn giáo viên đó vào cuối danh sách

Lưu ý: Ở phạm vi ngoài lớp chúng ta không thể truy xuất trực tiếp đến các thuộc tính (private) nên phải truy xuất thông qua phương thức ví dụ như phương thức getht().

|  |
| --- |
| GV gvtk;  **cin>>gvtk;**  bool KT;  **KT = false;**  int cs;  for(i = 0; i< n; i++ )  {  **if(strcmp(gvtk.getht(),A[i].getht())==0)**  {  **KT = true;**  **cs = i;**  }  }  **if**(**KT**)  {  cout<<"\nGiao vien nay co trong danh sach va co luong la: "  **<< A[cs].luong();**  }  **else**  {  cout<<"\nGiao vien nay khong co trong danh sach: ";  **n++;**  **A[n-1]=gvtk;**  } |

**Bài 4:**

Xây dựng lớp máy tính MAYTINH gồm các thuộc tính: Mã hàng, tên hãng sản xuất, đơn giá, Dung lượng bộ nhớ RAM, trọng lượng và lớp này gồm các hàm thành phần:

* Hàm khởi tạo với các tham số có giá trị ngầm định.
* Hàm hủy.
* Tái định nghĩa toán tử nhập “>>”.
* Hàm tính tiền vận chuyển. Biết nếu trọng lượng máy tính > 2 (đơn vị tính là kg) thì tiền vận chuyển là 100000, nếu trọng lượng máy tính > 1 (đơn vị tính là kg) thì tiền vận chuyển là 50000, còn lại là 30000.
* Tái định nghĩa toán tử xuất “<<”.

Nhập vào danh sách n máy tính (n nguyên dương nhập từ bàn phím), hiển thị danh sách máy tính có dung lượng bộ nhớ RAM bằng 16.

Nhập vào tên hãng sản xuất là SONY, nếu hãng này chưa có trong danh sách thì chèn hãng máy tính này vào đầu danh sách. Nếu có hãy xóa máy tính này khỏi danh sách?

**Hướng dẫn:**

* Xây dựng lớp MAYTINH với các thuộc tính và phương thức theo yêu cầu
* Tái định nghĩa toán tử nhập >>, xuất <<. Hai toán tử này nhất định phải được khai báo là hàm bạn của lớp MAYTINH.
* Trong chương trình chính tạo 1 mảng các đối tượng có kiểu MAYTINH. Các thao tác nhập, xuất, tìm kiếm, chèn, xóa được thực hiện tương tự như các thao tác trong những bài trước.

**Bài 5:**

Tạo lớp phân số PS gồm 2 thành phần thuộc tính là tử số và mẫu số kiểu số nguyên. Trong lớp có các hàm sau:

* Hàm khởi tạo có các tham số với giá trị ngầm định
* Hàm hủy
* Tái định nghĩa toán tử nhập >>, xuất <<.
* Tái định nghĩa các phép toán cộng +, trừ -, nhân \*, chia / 2 phân số.

Viết chương trình nhập vào 2 phân số, In ra màn hình tổng, hiệu, tích, thương của 2 phân số đó.

**Hướng dẫn:**

Tái định nghĩa toán tử cộng + (Toán tử + là hàm thành viên của lớp và là toán tử 2 ngôi nên số tham số của hàm toán tử này là 1). Định nghĩa hàm toán tử như sau:

|  |
| --- |
| **phanso operator +( phanso &p)**  {  phanso ps;  **ps.tuso = this->tuso\*p.mauso + p.tuso\*this->mauso;**  **ps.mauso = this->mauso\*p.mauso;**  return ps;  } |

Các toán tử -, \*, / định nghĩa hoàn toàn tương tự

Trong chương trình về phân số lưu ý nên rút gọn các phân số đó

Sau khi đã định nghĩa các toán tử +, - , \*, chia. Để thực hiện cộng, trừ, nhân, chia các đối tượng ta có thể viết trực tiếp như sau:

|  |
| --- |
| **phanso p,p1,p2;**  **p = p1+p2;**  **p = p1-p2;**  **p = p1 \* p2;**  **p = p1/p2;** |

Cài đặt chương trình như sau:

|  |
| --- |
| #include<iostream.h>  #include<math.h>  #include<iomanip.h>  #include<conio.h>  int UCLN(int a,int b)  {  a=abs(a);  b=abs(b);  while(a!=b)  {  if(a>b)  a=a-b;  else  b=b-a;  }  return a;  }  class PS  {  private:  float ts;  float ms;  public:  PS(float tuso=0,float mauso=1)  {  ts=tuso;  ms=mauso;  }  ~PS(){}  friend istream& operator >>(istream &is, PS &p)  {  cout<<"\nNhap tu so : "; is>>p.ts;  do  {  cout<<"\nNhap mau so : "; is>>p.ms;  if(p.ms == 0)  cout<<"\nnhap mau so khac 0, Moi ban nhap lai ! ";  }while(p.ms == 0);    return is;  }  friend ostream& operator <<(ostream &os, PS &p)  {  os<<p.ts<<"/"<<p.ms;  }  PS toigian()  {  PS p ;  p.ts = ts/UCLN(ts,ms);  p.ms = ms/UCLN(ts,ms);  return p;  }  PS operator +( PS &p)  {  PS ps;  ps.ts = this->ts\*p.ms + p.ts\*this->ms;  ps.ms = this->ms\*p.ms;    return ps.toigian();  }  PS operator -( PS &p)  {  PS ps;  ps.ts = (this->ts\*p.ms) - (p.ts\*this->ms);  ps.ms = this->ms\*p.ms;  return ps.toigian();  }  PS operator \*( PS &p)  {  PS ps;  ps.ts = this->ts\*p.ts;  ps.ms = this->ms\*p.ms;  return ps.toigian();  }  PS operator /( PS &p)  {  PS ps;  ps.ts = this->ts\*p.ms;  ps.ms = this->ms\*p.ts;  return ps.toigian();  }  };  int main()  {  PS p,p1,p2;  cout<<"\nNhap thong tin cho phan so 1 : ";  cin>>p1;  cout<<"\nNhap thong tin cho phan so 2 : ";  cin>>p2;  cout<<"\n\n\n - Cac phan so vua nhap : ";  cout<<p1;  cout<<" va ";  cout<<p2;  p = p1+p2;  cout<<"\n Tong = ";  cout<<p;  p = p1-p2;  cout<<"\n Hieu = ";  cout<<p;  p = p1 \* p2;  cout<<"\n tich = ";  cout<<p;  p = p1/p2;  cout<<"\n thuong = ";  cout<<p;  cout<<endl;  return 0;  } |

**Mở rộng:**

* Tạo lớp PS, định nghĩa các toán tử cộng +, trừ -, nhân \*. Viết chương trình nhập vào một dãy các PS và in ra màn hình tổng, hiệu, tích của các phân số đó.
* Định nghĩa toán tử so sánh lớn hơn >, <, <=, <= so sánh hai phân số

Viết chương trình:

* Nhập vào một dãy các PS và in ra màn hình tổng, hiệu, tích của các phân số đó.
* In ra phân số có giá trị lớn nhất.
* In ra phân số có giá trị nhỏ nhất

**Bài 6:**

Xây dựng lớp đa thức bậc 3 DaThuc3 gồm các thuộc tính hệ số bậc 3, bậc 2, bậc 1 và bậc 0 lần lượt là a, b, c, d (a, b, c, d nguyên) là các thuộc tính riêng. Trong lớp có định nghĩa các hàm sau:

* Hàm khởi tạo với các tham số có giá trị ngầm định bằng 0
* Hàm hiển thị đa thức ra màn hình ở dạng (ax3 + bx2 + cx + d )
* Hàm toán tử định nghĩa phép cộng, trừ 2 đa thức.

Viết chương trình nhập 2 đa thức và in ra màn hình tổng, hiệu 2 đa thức đó[1].

*Yêu cầu: Sinh viên vận dụng các kiến thức đã học tự hoàn thiện chương trình.*

**Bài 7:**

Xây dựng lớp ô tô OTO gồm các thuộc tính: Mã hàng, tên hãng sản xuất, đơn giá, dung tích xilanh, trọng lượng và lớp này gồm các hàm thành phần:

- Hàm khởi tạo với các tham số có giá trị ngầm định.

- Hàm hủy.

- Tái định nghĩa toán tử nhập “>>”.

- Hàm tính tiền vận chuyển. Biết nếu trọng lượng ô tô > 2 (đơn vị tính là tấn) thì tiền vận chuyển là 100000, nếu trọng lượng ô tô > 1 (đơn vị tính là tấn) thì tiền vận chuyển là 50000, còn lại là 30000.

- Tái định nghĩa toán tử bằng “= =” để so sánh dung tích xilanh ô tô với một số thực.

- Tái định nghĩa toán tử xuất “<<”.

Nhập vào danh sách n ô tô (n nguyên dương nhập từ bàn phím), hiển thị danh sách ô tô có dung tích xilanh bằng 2.5.

Nhập vào tên hãng sản xuất là TOYOTA, nếu hãng này chưa có trong danh sách thì chèn hãng ô tô này vào đầu danh sách. Nếu có hãy xóa ô tô này khỏi danh sách?

**Mở rộng:**

Với lớp OTO, hãy tái định nghĩa các toán tử so sánh >, >=, <, <= so sánh 2 đối tượng ô tô dựa trên 1 số thuộc tính.

**Hướng dẫn:**

Tái định nghĩa toán tử so sánh. Các toán tử so sánh sẽ có kết quả true hoặc false thuộc kiểu bool. Vì vậy kiểu trả về của các toán tử này có thể khai báo là kiểu bool, hoặc cũng có thể quy ước một kiểu khác.

Ví dụ: Tái định nghĩa toán tử bằng “= =” để so sánh dung tích xilanh 2 ô tô với nhau. Trong lớp ô tô định nghĩa phương thức sau:

|  |
| --- |
| bool **operator==(OTO** t)  {  if (**dt == t.dt**) // dt là thuộc tính dung tích xi lanh của oto  return **true**;  else return **false**;  } |

Trong chương trình có thể sử dụng toán tử bằng “==” để so sánh hai đối tượng ô tô với nhau

**Bài 8:**

Định nghĩa chồng phép gán “=” và phép truy xuất tới thành phần của mảng “[]” thao tác trên lớp vector.

Hướng dẫn:

|  |
| --- |
| #include<iostream.h>  #include<string.h>  **class VT**  {  int n;  float \*v;  public:  **VT ()**  {  int i;  cout<<"So chieu: ";  cin>>n;  v = new float[n];  }  **VT(VT &b);**  int length()  {  return n;  }  **VT& operator=(VT &b);**  **float& operator[](int i)**  {  **return v[i];**  }  **~VT();**  };  **VT::VT(VT &b)**  {  int i;  v = new float[b.n];  for(i=0;i<n;i++)  v[i]=b[i];  }  **VT::~VT()**  {  delete v;  }  **VT& VT:: operator=(VT &b)**  {  if(this!=&b) //  {  //xóa vùng nhớ động đã có đối tượng vế trái  **delete v;**  //cấp phát vùng nhớ mới có kích thước như trong đối tương b  **v = new float[b.n];**  for(int i =0; i<n;i++)  **v[i]=b.v[i];**  }  // Khi hai đối tượng giống nhau, không làm gì  **return \*this;**  }  void nhap(VT &s)  {  for (int i=0; i<s.length();i++)  {  cout<<"Toa do thu "<<i+1<<": ";  cin>>s[i];  }  }  void in(VT &s)  {  cout<<" so chieu: "<<s.length()<<endl;  for(int i=0;i<s.length();i++)  cout<<s[i]<<" ";  cout<<endl;  }  **int main()**  {  VT s1,s2;  cout<<"\nNhap cac toa do cua s1:"<<endl;  nhap(s1);  cout<<"\nIn thong tin ve s1: "<<endl;  in(s1);  **s2 = s1;**  cout<<"\nIn thong tin ve s2: "<<endl;  in(s2);  return 0;  } |

**Mở rộng:** Định nghĩa chồng các phép gán “=”, truy xuất tới thành phần mảng [] trên các lớp có thành phần dữ liệu động. Chẳng hạn lớp Ma trận có thể khai báo như sau:

|  |
| --- |
| **class MT**  **{**  **static int n,m;** //số hàng, số cột  **float \*\*m;** //vùng nhớ chứa các thành phần dữ liệu của ma trận  **}** |