# BÀI THỰC TẬP 07 – LẬP TRÌNH CÀI ĐẶT KHUÔN HÌNH

## A. MỤC TIÊU

Cung cấp cho sinh viên các kiến thức:

- Hiểu khái niệm khuôn hình hàm, khuôn hình lớp.

- Biết cách xây dựng chương trinh sử dụng khuôn hình hàm, khuôn hình lớp.

## B. NỘI DUNG

* Khai báo định nghĩa khuôn hình hàm, khuôn hình lớp.
* Sử dụng khuôn hình hàm, khuôn hình lớp triển khai bài toán thực tế.

## C. YÊU CẦU PHẦN CỨNG

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

## D. KẾT QUẢ SAU KHI HOÀN THÀNH

Hiểu và tự xây dựng được chương trình vận dụng kiến thức lập trình khuôn hình hàm, khuôn hình lớp.

## E. TÓM TẮT LÝ THUYẾT

**1. Khuôn hình hàm:**

Những hàm được xây dựng với những kiểu tham số trừu tượng do đó có thể sử dụng được với nhiều kiểu dữ liệu truyền vào khác nhau gọi là khuôn hình hàm. Chương trình dịch sẽ tự động tạo ra những thể hiện thích hợp nhất của hàm khi có lời gọi đến chúng trong chương trình.[1]

**2. Khuôn hình lớp:**

Những lớp được xây dựng trong đó các thành phần dữ liệu hoặc các hàm thành phần của lớp sử dụng các kiểu dữ liệu trừu tượng gọi làm khuôn hình lớp. Khi chương trình có nhu cầu sử dụng lớp với những kiểu dữ liệu cụ thể thì chương trình dịch sẽ tự động biên dịch những thể hiện thích hợp nhất.[1]

Có thể hình dung khuôn hình giống như cái khuôn đúc ra các hàm hoặc các lớp hoàn toàn giống nhau về mô hình xử lý, chỉ khác về kiểu dữ liệu áp dụng.

* ***Định nghĩa khuôn hình hàm:***

template <[tham số khuôn hình]>

[định nghĩa hàm]

Ví dụ:

template <class T>

T min(T a, T b)

{

return (a>b?b:a);

}

* ***Định nghĩa khuôn hình lớp:***

template <[tham số khuôn hình]>

[định nghĩa lớp]

Ví dụ:

template <class T, int n>

class array {

private:

T elements[n];

public:

T & operator[] (int i) {

return elements[i];

}

};

Trong đó: Tham số khuôn hình:

*- Tham số kiểu:* Đại diện cho một kiểu dữ liệu trừ tượng. Một khuôn hình hàm hay khuôn hình lớp có thể sử dụng được nhiều tham số kiểu. Một ví dụ của tham số biểu thức là kiểu T trong ví dụ khuôn hình hàm và khuôn hình lớp đã đưa ra ở trên.

*-* *Tham số biểu thức:* Đối với khuôn hình lớp, tham số biểu thức được sử dụng để tạo ra những lớp thể hiện linh động hơn. Giá trị truyền cho tham số biểu thức phải là các hằng số. Một ví dụ cho tham số biểu thức là tham số n trong ví dụ về khuôn hình lớp đã xét ở trên.

**3. Sử dụng khuôn hình hàm**

Khuôn hình hàm được sử dụng như một hàm thông thường. Khi thực hiện dựa trên kiểu dữ liệu của các tham số truyền vào cho hàm, chương trình dịch sẽ tạo ra một hàm thể hiện tương ứng.

**Ví dụ:**

main() {

int i,j;

float x,y;

….…

cout<<min(i,j);//tương ứng với hàm thể hiện min(int,int)

cout<<min(x,y);//tương ứng với hàm thể hiện min(float,float)

………

}

* ***Cụ thể hóa một hàm thể hiện***

//Thể hiện min (char \*,char \*) cho khuông hình hàm min

char \* min (char \* a, char \* b)

{

if (strcmp(a,b)<0) return a;

else return b;

};

* ***Định nghĩa chồng khuôn hình hàm***

Cũng giống như những định nghĩa chồng hàm, các khuôn hình hàm cũng có thể định nghĩa chồng.

template <class T>

T min (T a, T b)

{

return (a>b?b:a);

}

template <class T>

T min (T a, T b, T c)

{

if (a>b) return (b>c?c:b);

else return (a>c?c:a);

};

**4. Sử dụng khuôn hình lớp**

Khi định nghĩa các biên thuộc kiểu thể hiện của một khuôn hình lớp, cần phải chỉ rõ những kiểu dữ liệu cụ thể nào tương ứng các tham số kiểu hoặc hằng số nào tương ứng các tham số biểu thức.

main()

{

array<int,100> a;

…

for (i=0;i<100;i++)

cout <<a[i] <<``\n’’;

….

};

* **Cụ thể hóa các thành phần hàm trong một khuôn hình lớp**

Cũng giống như các thể hiện của khuôn hình hàm, đôi khi mô hình xử lý của một khuôn hình lớp đòi hỏi phải được xây dựng một cách đặc biệt đối với một số tham số kiểu. Khi đó, C++ hỗ trợ cách cụ thể hóa những thành phần hàm trong thể hiện tương ứng.

Ví dụ:

template <class T>

class Point

{

private :

T x,y;

public :

…

void print() {

cout <<”(“<<x<<”,”<<y<<”)\n”;

}

….

};

void point<char>::print()

{

cout<<”(“<< (int)x <<”,”<<(int)y <<”)\n”;

}

* **Khai báo bạn bè với lớp khuôn hình**

Khi một hàm hoặc một lớp được khai báo là bạn bè của một khuôn hình lớp thì nó sẽ là bạn bè của tất cả các lớp thể hiện của khuôn hình lớp đó.

Nếu như ta muốn hàm, lớp bạn bè là những thể hiện của khuôn hình hàm hay khuôn hình lớp thì trong khai báo bạn bè cần phải chỉ rõ những tham số khuôn hình cần thiết cho các thể hiện này.

teamplate <class T>

class A {

……

friend void f(A<T> );

friend class B<T>;

…..

};

## F. HƯỚNG DẪN CHI TIẾT

**Bài 1:** Xây dựng khuôn hình hàm tìm giá trị lớn nhất trong ba giá trị đầu vào?[1]

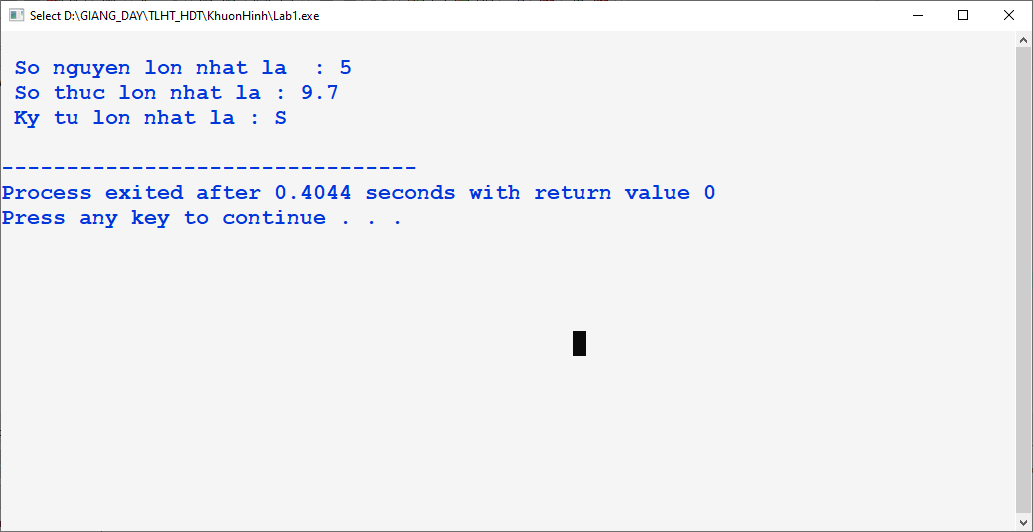
**- Bước 1:** Định nghĩa khuôn hình hàm tìm max cho 3 giá trị đầu vào

|  |
| --- |
| **#include<iostream.h>**  // Dinh nghia khuon hinh ham tim gia tri lon nhat  **template<class T>**  **T MAX( T a, T b, T c)**  {  T max=a;  if(b>a) max= b;  if(c>max) max=c;  return max;  } |

**- Bước 2:** Gọi khuôn hình hàm đã viết và sử dụng. Cú pháp gọi khuôn hình hàm giống gọi hàm bình thường.

|  |
| --- |
| **main()**  {  int a=3,b=4,c=5;  cout<<"\n So nguyen lon nhat la : "<< MAX(a,b,c);  float x=9.7, y= 7.6, z= 8;  cout<<"\n So thuc lon nhat la : "<< MAX(x,y,z);  char m='S', n='G', p='B';  cout<<"\n Ky tu lon nhat la : "<< MAX(m,n,p)<<endl;  } |

**Kết quả thực hiện chương trình:**



**Bài 2:** Cho mảng một chiều, n phần tử nguyên. Xây dựng khuôn hình hàm sau:

- Nhập mảng một chiều.

- Xuất thông tin trên mảng.

- Tính tổng mảng.

Viết chương trình sử dụng khuôn hình hàm đã viết

**Hướng dẫn:**

**Bước 1:** Khai báo các thư viện cần dùng.

#include<iostream.h>

**Bước 2:** Định nghĩa khuôn hình hàm nhập mảng. Tham số khuôn hình trong hàm này được đặt là T. T là một kiểu dữ liệu trừu tượng đại diện cho kiểu dữ liệu của mảng.

|  |
| --- |
| **template<class T>**  **void nhap( T a[], int n)**  {  for( int i=0;i<n;i++)  {  cout<<"a["<<i<<"]=";  cin>>a[i];  } |

**- Bước 3:**Tương tự định nghĩa khuôn hình hàm xuất mảng

|  |
| --- |
| // Dinh nghia khuon hinh ham xuat mang  **template<typename T>**  **void xuat(T a[], int n)**  {  for( int i=0;i<n;i++)  cout<<a[i]<<"\t";  } |

**Bước 4*:*** Định nghĩa khuôn hình hàm tính tổng phần tử trên mảng. Hàm tính tổng có kiểu dữ liệu là T vì các phần tử trên mảng có kiểu T.

|  |
| --- |
| **template<typename T>**  **T Tong(T \*a , int soPhanTu)**  {  T tong = 0;  for(int i = 0; i < soPhanTu; i++)  {  tong += a[i];  }  return tong;  } |

**Bước 5:**Định nghĩa hàm main() sử dụng khuôn hình hàm đã viết.

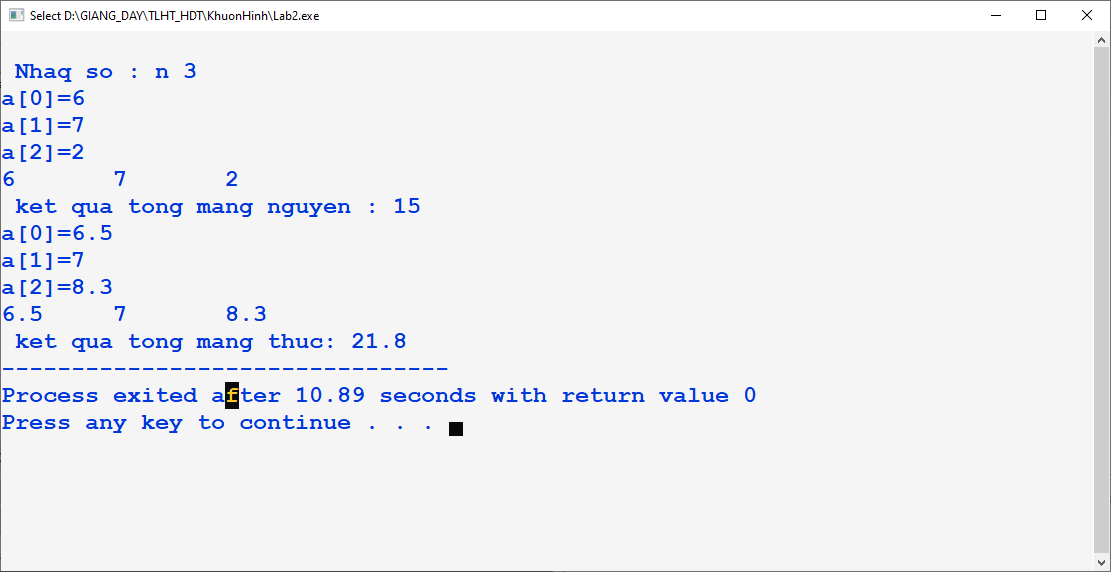
Lời gọi đến khuôn hình hàm là: Tong(a,n); Tong(b,n);

|  |
| --- |
| **main()**  {  int a[30], n;  cout<<"\n Nhap so : n "; cin>> n;  nhap(a,n); // goi ham  xuat(a,n);  cout<<"\n ket qua tong mang nguyen : "<<  Tong(a,n)<<endl; // goi khuon hinh ham  float b[30];  nhap(b,n);  xuat(b,n);  cout<<"\n ket qua tong mang thuc: "<<  Tong(b,n); // goi khuon hinh ham  } |

Chương trình hoàn chỉnh:

|  |
| --- |
| **#include<iostream.h>**  // Dinh nghia khuon hinh ham nhap mang  **template<class T>**  **void nhap( T a[], int n)**  {  for( int i=0;i<n;i++)  {  cout<<"a["<<i<<"]=";  cin>>a[i];  }  }  // Dinh nghia khuon hinh ham xuat mang  **template<typename T>**  **void xuat(T a[], int n)**  {  for( int i=0;i<n;i++)  cout<<a[i]<<"\t";  }  **template<typename T>**  **T Tong(T \*a , int soPhanTu)**  {  T tong = 0;  for(int i = 0; i < soPhanTu; i++)  {  tong += a[i];  }  return tong;  }  **main()**  {  int a[30], n;  cout<<"\n Nhaq so : n "; cin>> n;  nhap(a,n); // goi ham  xuat(a,n);  cout<<"\n ket qua tong mang nguyen : "<<  Tong(a,n)<<endl;  float b[30];  nhap(b,n);  xuat(b,n);  cout<<"\n ket qua tong mang thuc: "<<  Tong(b,n);  } |

Kết quả thực hiện chương trình:



**Bài 3:** Định nghĩa khuôn hình hàm nhập, xuất, tìm phần tử nhỏ nhất trên mảng. Viết chương trình nhập vào một mảng n phân số, tìm và in ra phân số nhỏ nhất?

**Hướng dẫn:**

**- Bước 1:**  Khai báo thư viện cần dùng.

#include<iostream.h>

**- Bước 2:** Theo yêu cầu đề bài cần tìm phân số nhỏ nhất trong danh sách mảng phân số, do đó cần bổ sung thêm một kiểu dữ liệu mới trong chương trình là kiểu phân số (ps). Bước tạo lớp phân số đã được hướng dẫn trong bài thực tập 02.

**- Bước 3:** Định nghĩa khuôn hình hàm min tìm phần tử nhỏ nhất với kiểu dữ liệu trừu tượng T

|  |
| --- |
| **template<typename T>**  **T Min(T \*a, int n)**  {  T min = a[0];  for(int i=0;i<n;i++)  if(min > a[i])  min = a[i];  return min;  } |

**- Bước 4:** Định nghĩa hàm main()

|  |
| --- |
| **int main()**  {  ps a[10];  int n;  cout<<"\n Nhap n : " ; cin>>n;  for(int i=0;i<n;i++){  cout<<"\nnhap phan so thu "<<i+1<<"\n";  a[i].Nhap();  }  cout<<"\n ------Danh sach phan so da nhap-----";  for(int i=0;i<n;i++){  cout<<"\n Phan so thu "<<i+1<<" : ";  a[i].Xuat();  }  // Tim nho nhat  cout<<"\nPhan so nho nhat la: "; Min(a,n).Xuat();  } |

**Mở rộng:** Cài đặt khuôn hình hàm sắp xếp mảng tăng dần sử dụng các thuật toán Selection sort, Insertion sort, Bubble sort.

**Bài 4:** Xây dựng khuôn hình lớp tìm giá trị lớn nhất trong hai giá trị đầu vào?

**Bước 1: Khai báo các thư viện cần dùng.**

#include<iostream.h>

**Bước 2: Định nghĩa khuôn hình lớp như sau:**

|  |
| --- |
| **template <class T>**  **class MyPair** {  private:  T a, b;  public:  MyPair(T first, T second) {  a = first;  b = second;  }  void Print()  {  cout << "First = " << a << ", Second = "<< b << endl;  }  T GetMax();  **};** |

**Bước 3: Định nghĩa hàm main():**

|  |
| --- |
| **main(){**  MyPair<int> pair1(100, 200);  pair1.Print();  cout <<"Max: "<< pair1.GetMax()<< endl;  MyPair<char> pair2('C', 'B');  pair2.Print();  cout << "Max: " << pair2.GetMax() << endl;  **}** |

**Bài 5:** Sử dụng khuôn hình lớp và định nghĩa các hàm cộng, trừ, nhân, chia, so sánh lớn hơn với đầu vào là hai giá trị có kiểu dữ liệu cơ bản.

**Hướng dẫn:**

**- Bước 1:** Định nghĩa khuôn hình lớp như sau:

- Thành phần dữ liệu riêng là hai biến đầu vào có kiểu trừu tượng T.

- Hàm thành phần bao gồm: Hàm tạo, hàm xuất, hàm tính tổng, hiệu, tích, thương, hàm tìm lớn nhất trong hai giá trị.

|  |
| --- |
| **#include <iostream>**  using namespace std;  **template <class T>**  **class Calculator**  {  **private:**  T num1, num2;  **public:**  Calculator(T n1, T n2)  {  num1 = n1;  num2 = n2;  }  **void displayResult()**  {  cout <<"Hai so la: "<<num1<< " va " << num2<< "."<<endl;  cout << "Cong: " << add() << endl;  cout << "Tru: " << subtract() << endl;  cout << "Nhan: " << multiply() << endl;  cout << "Chia: " << divide() << endl;  cout << "So lon nhat: " << GetMax() << endl;  }  **T add()** { return num1 + num2; }  **T subtract()** { return num1 - num2; }  **T multiply()** { return num1 \* num2; }  **T divide()** { return num1 / num2; }  **T GetMax()**  {  return num1 > num2 ? num1 : num2;  }  }; |

**- Bước 2**: Viết hàm main(): Sinh viên tự thực hiện

**Mở rộng:** Cài đặt bài toán với kiểu phân số, số phức.

**Bài 6:** Xây dựng khuôn hình lớp thực hiện các thao tác: tạo mảng, in mảng, tính tổng các phần tử trên mảng một chiều.

**Hướng dẫn:**

**- Bước 1:** Xây dựng khuôn hình lớp thực hiện các thao tác: tạo mảng, in mảng, tính tổng mảng.

|  |
| --- |
| #include <iostream>  #include <iomanip.h>  using namespace std;  **template <typename T>**  **class Array**  {  **private:**  T \*ptr;  int size;  **public:**  **Array(T arr[], int s);**  **void print();**  **void tong();**  }; |

Định nghĩa cụ thể từng hàm đã khai báo trong lớp

- Định nghĩa hàm print()

|  |
| --- |
| **template <typename T>**  **void Array<T>::print()**  {  for (int i = 0; i < size; i++)  cout<<setw(5)<<\*(ptr + i);  cout<<endl;  } |

- Định nghĩa hàm tính tổng tong()

|  |
| --- |
| **template <typename T>**  **void Array<T>::tong()**  {  T s=0;  for (int i = 0; i < size; i++)  s=s+ \*(ptr + i);  cout<<s<<endl;  } |

**- Bước 2:** Sử dụng khuôn hình lớp đã viết

|  |
| --- |
| **int main()**  {  // Tao mang so thuc  float arr[5] = {3, 2, 1.6, 4, 5};  Array<float> a(arr, 5);  cout<<"\nMang cac so thuc: ";  a.print();  cout<<"Tong mang :"; a.tong();  // Tao mang so nguyen  int arr1[5] = {5, 2, 6 , 4, 7};  Array<int> b(arr1, 5);  cout<<"\nMang cac so nguyen: ";  b.print();  cout<<"Tong mang :"; b.tong();  return 0;  } |

**Mở rộng:** tương tự, trong lớp Array bổ sung thêm các hàm: sắp xếp, tìm phần tử lớn nhất, nhỏ nhất, tính trung bình phần tử, tìm phần tử là số nguyên tố trên mảng (với mảng số nguyên), kiểm tra mảng đối xứng….. Sử dụng những hàm đã viết.

**Bài 7:** Xây dựng khuôn hình lớp mô tả thao tác push, pop phần tử trên cấu trúc dữ liệu stack?

**Hướng dẫn:**

**- Bước 1:** Tạo khuôn hình lớp mô tả chức năng stack sử dụng kiểu dữ liệu T.

Trong lớp khai báo các thành phần dữ liệu và hàm thành phần:

+ Mảng chứa các phần tử của ngăn xếp có kiểu T

+ Số phần tử: size

+ Chỉ số phần tử ở đỉnh danh sách: top

+ Hàm tạo

+ Hàm hủy

+ Hàm kiểm tra rỗng: isEmpty()

+ Hàm kiểm tra đầy isFull()

+ Hàm lấy ra một phần tử: pop()

+ Hàm đẩy vào một phần tử: push()

|  |
| --- |
| #include <iostream>  using namespace std ;  template <class T>  class Stack  {  public:  Stack(int = 10) ; // Khoi tao co 10 phan tu.  ~Stack()  {  delete [] stackPtr ;  }  int push(const T&);  int pop(T&) ; // lay mot phan tu khoi stack  int isEmpty()const { return top == -1 ; }  int isFull() const { return top == size - 1 ; }  private:  int size ; // So phan tu trong stack  int top ;  T\* stackPtr ;  } ; |

Định nghĩa cụ thể những hàm đã khai báo trong lớp.

- Định nghĩa hàm tạo

|  |
| --- |
| Dinh nghia cac ham trong lop  //khoi tao mac dinh kich thuoc 10 phan tu  template <class T>  Stack<T>::Stack(int s)  {  size = s > 0 && s < 1000 ? s : 10 ;  top = -1 ; // khoi tao stack  stackPtr = new T[size] ;  } |

- Định nghĩa hàm push đẩy vào một phần tử:

|  |
| --- |
| template <class T>  int Stack<T>::push(const T& item)  {  if (!isFull())  {  stackPtr[++top] = item ;  return 1 ;  }  return 0 ;  } |

- Định nghĩa hàm lấy ra một phần tử:

|  |
| --- |
| // lay mot phan tu khoi stack  template <class T>  int Stack<T>::pop(T& popValue)  {  if (!isEmpty())  {  popValue = stackPtr[top--] ;  return 1 ;  }  return 0 ;  } |

**- Bước 2:** Viết hàm main() sử dụng khuôn hình hàm đã định nghĩa.

|  |
| --- |
| main()  {  typedef Stack<float> FloatStack ;  typedef Stack<int> IntStack ;  FloatStack fs(5) ;  float f = 1.2;  cout<<"\n ---- Quan ly cac so thuc------";  cout << "\nDay du lieu vao Stack" << endl ;  while (fs.push(f))  {  cout << f << ' ' ;  f += 1.1 ;  }  cout << endl << "Stack day!!!" << endl  << endl << "Lay du lieu tu Stack " << endl ;  while (fs.pop(f))  cout << f << ' ' ;  cout << endl << "Stack rong!!!" << endl ;  cout << endl ;  } |

**Bài tập tự giải:**

**Bài 8:** Xây dựng các khuôn hình hàm thực hiện thao tác trên mảng một chiều:

**-** Tìm phần lớn nhất trên mảng

**-** Tính tổng các phần tử.

- Đổi chỗ hai phần tử.

- Sắp xếp các phần tử theo chiều tăng dần.

- Đếm phần tử khác một giá trị cho trước.

- Tìm phần tử theo giá trị X được nhập vào.

- Xóa phần tử trên mảng tại vị trí k

- Thêm phần tử tại vị trí k.

Hãy sử dụng các khuôn hình hàm đã viết thực hiện trên kiểu dữ liệu phân số, và số phức.

**Bài 9:** Xây dựng các khuôn hình hàm thực hiện thao tác trên mảng hai chiều:

**-** Tìm phần lớn nhất trên mảng

**-** Tính tổng các phần tử trên đường chéo chính

- Sắp xếp các phần tử trên đường chéo theo chiều tăng dần.

Hãy sử dụng các khuôn hình hàm đã viết thực hiện trên kiểu dữ liệu phân số, và số phức.

**Bài 10:** Xây dựng khuôn hình lớp thực hiện các thao tác:

* Tạo mảng
* Xuất mảng.
* Tính tổng các phần tử trên mảng.
* Tìm phần tử nhỏ nhất
* Sắp xếp danh sách phần tử giảm dần.

Áp dụng khuôn hình đã xây dựng tạo danh sách Nhân viên. Biết lớp Nhân viên được định nghĩa gồm các thông tin: mã, họ tên, hệ số lương, ngày công.

**Bài 11:** Xây dựng khuôn hình lớp mô tả thao tác push, pop trên stack. Áp dụng quản lý các phần tử có kiểu số nguyên, số thực, máy tính (Sinh viên tự xây dựng lớp MayTinh với các thông tin: Tên hãng, màu sắc, dung lượng, tốc độ)