**Bài tập cấu trúc dữ liệu và giải thuật**

1. **Cơ bản về C++**

**Bài 1.** Dùng hàm cin, cout để nhập vào 3 số nguyên kiểm tra xem nó có thể tạo thành 3 cạnh của tam giác vuông không?

|  |
| --- |
| #include <iostream>  #include <algorithm>  using namespace std;  bool CheckTriangle90(int A[3]) {  if (A[0] < A[1]) swap(A[0], A[1]);  if (A[0] < A[2]) swap(A[0], A[2]);  if (A[1] < A[2]) swap(A[1], A[2]);  return (A[0] \* A[0] == A[1] \* A[1] + A[2] \* A[2]);  }  int main()  {  int A[3];  cout << "Nhap vao 3 so nguyen : ";  cin >> A[0] >> A[1] >> A[2];  if (CheckTriangle90(A)) {  cout << "Co the tao 1 tam giac vuong" << endl;  }  else {  cout << "Khong the" << endl;  }  return 0;  } |

**Bài 2.** Nhập vào xâu ký tự (kiểu string) bằng cin xuất ra xâu viết hoa và xâu viết thường của xâu đó

|  |
| --- |
| #include <iostream>  #include <algorithm>  #include <string>  using namespace std;  string Upcase(string text) {  string res = text;  for (int i = 0; i < res.length(); i++) {  if (res[i] >= 'a' && res[i] <= 'z') {  res[i] -= 'a' - 'A';  }  }  return res;  }  string Lowcase(string text) {  string res = text;  for (int i = 0; i < res.length(); i++) {  if (res[i] >= 'A' && res[i] <= 'Z') {  res[i] -= 'A' - 'a';  }  }  return res;  }  int main()  {  string text;  cin >> text;  cout << "In hoa : " << Upcase(text) << endl;  cout << "In thuong : " << Lowcase(text) << endl;  return 0;  } |

**Bài 3.** Viết hàm tìm max và min của 3 số nguyên theo nguyên mẫu dưới đây, sau đó nhập vào 3 số nguyên tìm giá trị lớn nhất và nhỏ nhất

**void maxmin(int a,int b,int c,int &Max, int &Min);**

|  |
| --- |
| #include <iostream>  #include <algorithm>  #include <string>  #include <limits.h>  using namespace std;  void maxmin(int a, int b, int c, int& Max, int& Min) {  if (a > b)  swap(a, b);  if (a > c)  swap(a, c);  if (b > c)  swap(b, c);  Max = c;  Min = a;  }  int main()  {  int a, b, c, maxValue, minValue;  cin >> a >> b >> c;  maxValue = INT\_MIN;  minValue = INT\_MAX;  maxmin(a, b, c, maxValue, minValue);  cout << "Max : " << maxValue << endl;  cout << "Min : " << minValue << endl;  return 0;  } |

**Bài 4.** Viết hàm có đối mặc định để có thể vừa tìm max vừa tìm min tùy theo đối ismax theo nguyên mẫu dưới đây, sau đó áp dụng nhập vào 3 số nguyên khác nhau tìm số không phải max, min bằng cách tính tổng rồi trừ đi max và min

**int Sub(int a,int b,bool ismax=true);**

|  |
| --- |
| #include <iostream>  #include <algorithm>  #include <string>  using namespace std;  int Sub(int a, int b, bool ismax = true) {  // true = max  // false = min  if (ismax)  if (a > b) return a;  else  return b;  else  if (a > b)  return b;  else  return a;  }  int main()  {  int a, b, c, maxValue, minValue;  cout << "Nhap 3 so nguyen : ";  cin >> a >> b >> c;  maxValue = Sub(a, Sub(b, c));  minValue = Sub(a, Sub(b, c, false), false);  cout << "Gia tri con lai : "  << (a + b + c - maxValue - minValue) << endl;  return 0;  } |

**Bài 5.** Viết hàm mẫu (template) tìm max của hai số, áp dụng tìm max của 2 số nguyên và hai số thực.

|  |
| --- |
| #include <iostream>  using namespace std;  template<typename T> T Max(T a, T b) {  if (a > b) return a;  else  return b;  }  int main()  {  int a, b;  float c, d;  cout << "Nhap 2 so nguyen : ";  cin >> a >> b;  cout << "max integer number = " << Max(a,b) << endl;  cout << "Nhap 2 so thuc : ";  cin >> c >> d;  cout << "max real number = " << Max(c, d) << endl;  return 0;  } |

**Bài 6.**Viết hàm mẫu (template) tìm ước chung lớn nhất của 2 số, và hàm mẫu tìm ước chung lớn nhất của một dãy, áp dụng nhập vào một dãy kiểu long tìm ước chung lớn nhất của dãy.

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  template<typename T> T gcd(T a, T b) {  if (b == 0)  return a;  else  return gcd(b, a % b);  }  template<typename T> T gcdArray(T arr[], T n) {  T i;  T res = abs(arr[0]);  for (i = 1; i < n; i++) {  res = gcd(res, abs(arr[i]));  }  return res;  }  int main()  {  long arr[100];  long n;  cout << "Nhap so luong phan tu : ";  cin >> n;  for (long i = 0; i < n; i++) {  cout << "A[" << i << "] = ";  cin >> arr[i];  }  cout << "UCLN cua mang : " << gcdArray(arr, n);  return 0;  } |

1. **Lập trình hướng đối tượng trong C++**

**Bài 7.** Khai báo lớp điểm gồm 2 tọa độ và hàm nhập, và lớp đa giác gồm số đỉnh và con trỏ điểm, có các phương thức nhập các đỉnh, tính chu vi, tính điện tích đa giác. Viết hàm main mô phỏng chương trình

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  class point {  private:  double \_x, \_y;  public:  point();  point(double, double);  ~point();  void input();  void display();  double getX();  double getY();  friend double distance(const point& a, const point& b);  };  point::point() { ; }  point::point(double a, double b) {  this->\_x = a;  this->\_y = b;  }  point::~point() { ; }  void point::input() {  cin >> this->\_x >> this->\_y;  }  void point::display() {  cout << "(" << this->\_x << ", " << this->\_y << ")";  }  double point::getX() {  return this->\_x;  }  double point::getY() {  return this->\_y;  }  double distance(const point& a, const point& b) {  return sqrt(pow(a.\_x - b.\_x, 2) + pow(a.\_y - b.\_y, 2));  }  class polygon {  private:  int n;  point\* arr;  public:  polygon();  ~polygon();  void input();  void push\_point(point A);  void display();  double P();  double S();  };  polygon::polygon() { ; }  polygon::~polygon() { ; }  void polygon::input() {  cin >> this->n;  arr = new point[n];  for (int i = 0; i < n; i++) {  arr[i].input();  }  }  void polygon::push\_point(point A) {  arr = new point[n + 1];  this->n++;  arr[n - 1] = A;  }  double polygon::P() {  double res = 0;  for (int i = 0; i < this->n - 1; i++) {  res += distance(arr[i], arr[i + 1]);  }  return res;  }  double polygon::S() {  int j = this->n - 1;  double res = 0;  for (int i = 0; i < this->n; i++) {  res += (arr[j].getX() + arr[i].getX()) \* (arr[j].getY() - arr[i].getY());  }  return abs(res / 2);  }  int main()  {  polygon P1;  P1.input();  cout << "dien tich : " << P1.S() << endl;  cout << "chu vi : " << P1.P();  return 0;  } |

**Bài 8.**Khai báo lớp đa thức với bậc và mảng các hệ số, viết các phương thức nhập và xuất, tính giá trị của đa thức tại giá trị t. Viết hàm main thực hiện chương trình

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  class DT {  private:  int n; // level  float\* arr; // he so  public:  DT();  DT(int len);  ~DT();  void input();  void display();  double Fx(double x);  };  DT::DT() { ; }  DT::DT(int len) {  this->n = len;  arr = NULL;  }  DT::~DT() { ; }  void DT::input() {  cout << "Nhap so luong phan tu : ";  cin >> this->n;  arr = new float[n];  for (int i = 0; i <= n; i++) {  cin >> this->arr[i];  }  }  void DT::display() {  int curr = n;  cout << endl << "DT = ";  for (int i = 0; i <= n; i++) {  if (i != n)  cout << this->arr[i] << "x" << "^" << curr-- << " + ";  else  cout << this->arr[i];  }  }  double DT::Fx(double x) {  int curr = this->n;  double res = 0;  for (int i = 0; i <= n; i++) {  res += this->arr[i] \* pow(x, curr--);  }  return res;  }  int main()  {  DT dt1;  dt1.input();  dt1.display();  double x;  cout << endl << "x = ";  cin >> x;  double res = dt1.Fx(x);  cout << endl << "Fx(" << x << ") = " << res;  return 0;  } |

**Bài 9.**Khai báo lớp ma trận nguyên vuông gồm cỡ ma trận và con trỏ 2 chiều chứa các phần tử của ma trận (int \*\*a), xây dựng các phương thức nhập, xuất, chuyển vị, kiểm tra tam giác trên, tính tổng, tìm phần tử lớn nhất. Viết hàm main mô phỏng

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  class matrix {  private:  int level;  int\*\* a;  public:  matrix();  matrix(const matrix &p);  ~matrix();  void input();  void display();    int Sum();  int maxValue();  matrix chuyenVi();  };  matrix::matrix() { ; }  matrix::matrix(const matrix& p) {  \*this = p;  }  matrix::~matrix() { ; }  void matrix::input() {  cin >> this->level;  this->a = new int\*[this->level];  for (int i = 0; i < this->level; i++) {  a[i] = new int[this->level];  for (int j = 0; j < this->level; j++) {  cin >> this->a[i][j];  }  }  }  void matrix::display() {  for (int i = 0; i < this->level; i++) {  for (int j = 0; j < this->level; j++) {  cout << this->a[i][j] << " ";  }  cout << endl;  }  }  int matrix::Sum() {  int res = 0;  for (int i = 0; i < this->level; i++) {  for (int j = 0; j < this->level; j++) {  res += this->a[i][j];  }  }  return res;  }  int matrix::maxValue() {  int M = this->a[0][0];  for (int i = 0; i < this->level; i++) {  for (int j = 0; j < this->level; j++) {  M = max(M, this->a[i][j]);  }  }  return M;  }  matrix matrix::chuyenVi() {  matrix res;  res.a = new int\* [this->level];  for (int i = 0; i < this->level; i++) {  res.a[i] = new int[this->level];  }  res.level = this->level;  for (int i = 0; i < this->level; i++) {  for (int j = 0; j < this->level; j++) {  res.a[j][i] = this->a[i][j];  }  }  return res;  }  int main()  {  matrix M1;  M1.input();  M1.display();  cout << "Tong ma tran : " << M1.Sum() << endl;  cout << "Gtri lon nhat : " << M1.maxValue() << endl;  cout << "Matrix chuyen vi : " << endl;  (M1.chuyenVi()).display();  return 0;  } |

**Bài 10.**Khai báo lớp điểm có hàm tạo không đối gán tọa độ (0,0) và hàm tọa có đối gán tọa độ (a,b), phương thức xuất, tính diện tích hình tròn có đường kính đi qua gốc tọa độ và điểm hiện tại. Viết hàm main mô phỏng

|  |
| --- |
| #include <iostream>  #include <string>  #include <algorithm>  #include <vector>  #include<math>  using namespace std;  const double PI = 3.14159265359;  class Point {  double \_x, \_y;  public:  Point() {  \_x = \_y = 0;  }  Point(double a, double b) {  \_x = a;  \_y = b;  }  ~Point() { ; }  void Set(double x, double y) {  \_x = x;  \_y = y;  }  void input() {  cin >> this->\_x >> this->\_y;  }  void Display() {  cout << "(" << \_x << ", " << \_y << ")";  }  int getX() {  return \_x;  }  int getY() {  return \_y;  }  friend bool operator!=(const Point a, const Point b);  friend double distance(const Point& a, const Point& b);  };  double distance(const Point& a, const Point& b) {  return sqrt(pow(a.\_x - b.\_x, 2) + pow(a.\_y - b.\_y, 2));  }  int main() {  Point p1;  p1.input();  double R = distance(Point(), p1);  double res = R \* R \* PI;  cout << "Dien tich hinh trong tam O(0,0) di qua diem";  p1.Display();  cout << " la : " << res;  return 0;  } |

**Bài 11**. Khai báo lớp đa thức có hệ số là con trỏ chưa cấp phát (float \*a) viết hàm tạo có đối số là bậc của đa thức, cấp phát bộ nhớ động, viết hàm hủy giải phóng bộ nhớ động, phương thức nhập, xuất đa thức.

|  |
| --- |
| #include <iostream>  using namespace std;  typedef long long ll;  class DT {  private:  float\* a;  int len;  public:  DT();  DT(int n);  ~DT();  void input();  void display();  double Calc(int x);  };  DT::DT() {  a = NULL;  }  DT::~DT() {  delete[] a;  }  DT::DT(int n) {  a = new float[n + 1];  len = n;  }  void DT::input() {  if (!this->len) {  cout << "Nhap so bac : ";  cin >> this->len;  a = new float[len + 1];  }  for (int i = 0; i <= len; i++) {  cin >> a[i];  }  }  void DT::display() {  int curr = len;  for (int i = 0; i <= len; i++) {  if (curr == 0)  cout << a[i];  else  cout << a[i] << ".x^" << curr << " + ";  curr--;  }  }  double DT::Calc(int x) {  int curr = len;  double res = 0;  for (int i = 0; i <= len; i++) {  res += a[i] \* pow(x, curr);  }  return res;  }  int main() {  int n;  cout << "Nhap bac cua da thuc : ";  cin >> n;  DT dt1(n);  cout << "Nhap he so cua da thuc : ";  dt1.input();  cout << "Da thuc va nhap : " << endl;  dt1.display();    cout << "\n\n\n" << "\t1. Tinh DT voi he so x.\n\t2. Nhap da thuc moi\n\t3. Thoat\n\n";  int c;  bool isQuit = false;  do {  cout << "\n\tLua chon : ";  cin >> c;  switch (c)  {  case 1:  int x;  cout << "Nhap x : ";  cin >> x;  cout << "Fx(" << x << ") = " << dt1.Calc(x) << endl;  break;  case 2:  dt1.~DT();  cout << "Nhap da thuc : " << endl;  dt1.input();  cout << "DT vua nhap : " << endl;  dt1.display();  break;  default:  isQuit = true;  break;  }  if (!isQuit)  {  cout << "\n\n\n" << "\t1. Tinh DT voi he so x.\n\t2. Nhap da thuc moi\n\t3. Thoat\n\n";  }  } while (!isQuit);  return 0;  } |

**Bài 12.** Xây dựng lớp biểu diễn các đối tượng dãy số với các phương thức hàm tạo, hàm in, hàm thêm một phần tử vào dãy, hàm xóa một phần tử của dãy, hàm tìm kiếm một phần tử có trong dãy không nếu có trả lại vị trí của phần tử đó trong dãy.

|  |
| --- |
| #include <iostream>  using namespace std;  struct Node {  int \_data;  Node\* \_next;  };  class List {  private:  Node\* root;  Node\* createNode(int value);  public:  List();  ~List();    void push(int value);  void erase(int k);  void pop();  void display();  int find(int value);  };  List::List() {  root = NULL;  }  List::~List() {  delete root;  }  Node\* List::createNode(int value) {  Node\* p = new Node();  p->\_data = value;  p->\_next = NULL;  return p;  }  void List::push(int value) {  if (root == NULL) {  root = createNode(value);  return;  }  else {  Node\* curr = root;  while (curr->\_next != NULL) curr = curr->\_next;  curr->\_next = createNode(value);  }  }  void List::pop() {  if (root != NULL) {  Node\* curr = root;  Node\* p = curr;  while (curr->\_next != NULL) {  p = curr;  curr = curr->\_next;  }  delete p->\_next;  p->\_next = NULL;  }  }  void List::erase(int k) {  // Erase Kth number  if (this->root == NULL)  return;  k--;  Node\* curr = this->root;  Node\* p = curr;  while (k > 0 && curr->\_next != NULL) {  p = curr;  curr = curr->\_next;  k--;  }  if (k == 0) {  p->\_next = curr->\_next;  }  }  void List::display() {  if (root == NULL) return;    Node\* curr = root;  while (curr != NULL) {  cout << curr->\_data << " ";  curr = curr->\_next;  }  cout << endl;  }  int List::find(int value) {  // The func return the first position, have value in list  if (root == NULL) return -1;  Node\* curr = root;  int k = 0;  while (curr->\_next != NULL && curr->\_data != value) {  curr = curr->\_next;  k++;  }  if (curr->\_data == value)  return k + 1;  else  return -1;  }  int main() {  List l1;  l1.push(1);  l1.push(2);  l1.push(3);  l1.push(4);  l1.display();  cout << l1.find(2);  return 0;  } |

**Bài 13.**Xây dựng lớp biểu diễn các đối tượng là các sinh viên (các thuộc tính, phương thức do sv tự xác định)

|  |
| --- |
| // bai 33  #include <iostream>  #include <vector>  #include <stdlib.h>  #include <string>  using namespace std;  class student {  private:  string \_fullName;  int \_age;  string \_idStudent;  string \_address;  float \_mark;  public:  student();  ~student();  void input();  void display();  friend void sort\_Student\_List(vector<student>& arr);  friend void swapStudent(student& A, student& B);  };  student::student() { ; }  student::~student() { ; }  string inSTRING() {  string res;  do {  getline(cin, res);  } while (res == "");  return res;  }  void student::input() {  cout << "\tTen : ";  this->\_fullName = inSTRING();  cout << "\tMa sv : ";  this->\_idStudent = inSTRING();  cout << "\tTuoi : ";  cin >> this->\_age;  cout << "\tDia chi : ";  this->\_address = inSTRING();  cout << "\tDiem : ";  cin >> this->\_mark;  }  void student::display() {  cout << "\t" << this->\_fullName;  cout << "\t" << this->\_age;  cout << "\t" << this->\_idStudent;  cout << "\t" << this->\_address;  cout << "\t" << this->\_mark;  cout << endl;  }  int main() {  int n;  cin >> n;  vector<student> arr(n);  for (int i = 0; i < n; i++) {  cout << "Nhap SV[" << i << "] : " << endl;  arr[i].input();  }  cout << "==================================================================================" << endl;  cout << "Danh sach sv vua nhap : " << endl;  for (int i = 0; i < n; i++) {  cout << i + 1;  arr[i].display();  }  cout << "-----------------------------------------------------------------------------------" << endl;  return 0;  } |

**Bài 14.** Xây dựng lớp biểu diễn tập hợp số nguyên có hàm tạo, hàm hủy, có hàm rút gọn những phần tử trùng nhau chỉ giữ lại 1 phần tử, hàm sắp xếp tăng dần

|  |
| --- |
| #include <iostream>  #include <vector>  #include <stdlib.h>  #include <sstream>  #include <string>  using namespace std;  class INTEGER {  string \_digits;  void handle\_digits() {  while (this->\_digits.size() > 0  && this->\_digits.at(0) == '0')  this->\_digits = this->\_digits.substr(1);  }  public:  // Constructor & Distructor  INTEGER();  ~INTEGER();  INTEGER(long long a);  INTEGER(string a);  // ----- Operator ----- //  friend void swap(INTEGER& a, INTEGER& b);  // compare  friend bool operator!=(const INTEGER& a, const INTEGER& b);  friend bool operator==(const INTEGER& a, const INTEGER& b);  friend bool operator<(const INTEGER& a, const INTEGER& b);  friend bool operator>(const INTEGER& a, const INTEGER& b);  // Input Output  friend istream& operator>>(istream& is, INTEGER& a);  friend ostream& operator<<(ostream& os, INTEGER& a);  // Operator  friend void operator++(INTEGER& a);  friend void operator--(INTEGER& a);  friend void operator+=(INTEGER& a, const INTEGER& b);  friend INTEGER operator+(const INTEGER& a, const INTEGER& b);  friend void operator\*=(INTEGER& a, const INTEGER& b);  friend INTEGER operator\*(const INTEGER& a, const INTEGER& b);  // Application  friend INTEGER factorial(int n);  friend void BB\_sort(vector<INTEGER>& arr);  // Add an element into a Set (Mang loai bo cac phan tu trung nhau)  friend void PUSH\_BACK(const INTEGER& a, vector<INTEGER>& arr);  friend void DISPLAY\_ARR(vector<INTEGER>& arr);  };  INTEGER::INTEGER() {  this->\_digits.push\_back('0');  }  INTEGER::INTEGER(long long a) {  stringstream ss;  ss << a;  string str = ss.str();  this->\_digits = str;  }  INTEGER::INTEGER(string a) {  this->\_digits = a;  }  INTEGER::~INTEGER() { ; }  void swap(INTEGER& a, INTEGER& b) {  INTEGER temp = a;  a = b;  b = temp;  }  bool operator!=(const INTEGER& a, const INTEGER& b) {  return a.\_digits != b.\_digits;  }  bool operator==(const INTEGER& a, const INTEGER& b) {  return (a.\_digits == b.\_digits);  }  bool operator<(const INTEGER& a, const INTEGER& b) {  if (a.\_digits.length() < b.\_digits.length()) return true;  else if (a.\_digits.length() > b.\_digits.length()) return false;  int len = a.\_digits.length();  for (int i = 0; i < len; i++) {  if (a.\_digits[i] < b.\_digits[i])  return true;  }  return false;  }  bool operator>(const INTEGER& a, const INTEGER& b) {  if (a.\_digits.length() > b.\_digits.length()) return true;  else if (a.\_digits.length() < b.\_digits.length()) return false;  int len = a.\_digits.length();  for (int i = 0; i < len; i++) {  if (a.\_digits[i] > b.\_digits[i])  return true;  }  return false;  }  bool isCurr(INTEGER value, vector<INTEGER> arr) {  for (int i = 0; i < arr.size(); i++) {  if (value != arr[i]);  else  return true;  }  return false;  }  istream& operator>>(istream& is, INTEGER& a) {  is >> a.\_digits;  return is;  }  ostream& operator<<(ostream& os, INTEGER& a) {  os << a.\_digits;  return os;  }  void operator++(INTEGER& a) {  int carry, temp;  carry = 0;  temp = a.\_digits[a.\_digits.length() - 1] - '0' + 1;  if (temp < 10) {  a.\_digits[a.\_digits.length() - 1] = temp + '0';  }  else {  carry = temp / 10;  temp %= 10;  a.\_digits[a.\_digits.length() - 1] = temp + '0';  }  for (int i = a.\_digits.length() - 2; i >= 0; i--) {  temp = a.\_digits[i] - '0' + carry;  if (temp < 10) {  a.\_digits[i] = temp + '0';  carry = 0;  break;  }  else {  carry = temp / 10;  temp %= 10;  a.\_digits[i] = temp + '0';  }  }  if (carry > 0) {  a.\_digits.insert(a.\_digits.begin(), '1');  }  }  void operator--(INTEGER& a) {  int carry = 0, temp;  temp = a.\_digits[a.\_digits.length() - 1] - '0' - 1;  if (temp < 0) {  a.\_digits[a.\_digits.length() - 1] = temp + 10 + '0';  carry = -1;  }  else {  a.\_digits[a.\_digits.length() - 1] = temp + '0';  }  for (int i = a.\_digits.length() - 2; i >= 0; i--) {  temp = a.\_digits[i] - '0' + carry;  if (temp < 0) {  a.\_digits[i] = temp + 10 + '0';  carry = -1;  }  else {  a.\_digits[i] = temp + '0';  carry = 0;  break;  }  }  a.handle\_digits();  }  void operator+=(INTEGER& a, const INTEGER& b) {  INTEGER b1 = b;  while (a.\_digits.length() < b1.\_digits.length())  a.\_digits.insert(a.\_digits.begin(), '0');  while (a.\_digits.length() > b1.\_digits.length())  b1.\_digits.insert(b1.\_digits.begin(), '0');  int temp, carry, len;  carry = 0;  len = a.\_digits.length();  for (int i = len - 1; i >= 0; i--) {  temp = a.\_digits[i] - '0' + b1.\_digits[i] - '0' + carry;  if (temp >= 10) {  carry = temp / 10;  a.\_digits[i] = temp % 10 + '0';  }  else {  carry = 0;  a.\_digits[i] = temp + '0';  }  }  if (carry > 0)  a.\_digits.insert(a.\_digits.begin(), '1');  }  INTEGER operator+(const INTEGER& a, const INTEGER& b) {  INTEGER a1 = a;  a1 += b;  return a1;  }  void operator\*=(INTEGER& a, const INTEGER& b) {  int temp, carry = 0, len1, len2;  len1 = a.\_digits.length();  len2 = b.\_digits.length();  INTEGER Value;  for (int i = len1 - 1; i >= 0; i--) {  INTEGER T;  T.\_digits.pop\_back();  for (int j = 0; j < len1 - 1 - i; j++) T.\_digits.push\_back('0');  for (int j = len2 - 1; j >= 0; j--) {  temp = (a.\_digits[i] - '0') \* (b.\_digits[j] - '0');  if (temp > 0)  {  carry = temp / 10;  }  else {  carry = 0;  }  T.\_digits.insert(T.\_digits.begin(), temp % 10 + '0');  }  Value += T;  }  if (carry > 0) {  Value.\_digits.insert(Value.\_digits.begin(), carry + '0');  }  a = Value;  a.handle\_digits();  }  INTEGER operator\*(const INTEGER& a, const INTEGER& b) {  INTEGER T = a;  T \*= b;  return T;  }  INTEGER factorial(int n) {  // F[i] = F[i - 1] + F[i - 2]  // a , b , c  if (n <= 2)  return INTEGER(1);  INTEGER a(1);  INTEGER b(1);  INTEGER c(2);  for (int i = 4; i <= n; i++) {  a = b;  b = c;  c = a + b;  }  return c;  }  void BB\_sort(vector<INTEGER>& arr) {  INTEGER T;  for (int i = 0; i < arr.size() - 1; i++) {  for (int j = i + 1; j < arr.size(); j++) {  if (arr[i] > arr[j])  swap(arr[i], arr[j]);  }  }  }  void DISPLAY\_ARR(vector<INTEGER>& arr) {  for (int i = 0; i < arr.size(); i++) {  cout << arr[i] << " ";  }  }  void PUSH\_BACK(const INTEGER& a, vector<INTEGER>& arr) {  int i;  INTEGER a1 = a;  for (i = 0; i < arr.size(); i++) {  if (arr[i] == a1)  return;  }    arr.push\_back(a1);  }  int main() {  int n;  cin >> n;  INTEGER value;  vector<INTEGER> arr;    for (int i = 0; i < n; i++) {  cin >> value;  PUSH\_BACK(value, arr);  }    cout << endl << "Buble sort array !!" << endl; // Put a message  BB\_sort(arr);  DISPLAY\_ARR(arr);    return 0;  } |

**Bài 15.** Xây dựng lớp mẫu phân số thực hiện nhập vào n số nguyên a1,... an tìm giá trị tối giản của phân số



|  |
| --- |
| #include <iostream>  using namespace std;  int \_gcd(int a, int b) {  if (b == 0) return a;  else  return \_gcd(b, a % b);  }  class PS {  private:  int \_tu, \_mau;  void Handle() {  int temp = \_gcd(this->\_tu, this->\_mau);  this->\_tu /= temp;  this->\_mau /= temp;  }  public:  PS();  PS(int a, int b);  ~PS();  void set(int a, int b);  void input();  int getTu() {  return this->\_tu;  }  int getMau() {  return this->\_mau;  }  /\* Operator \*/  friend void operator+=(PS& A, const PS& B);  friend PS operator+(const PS& A, const PS& B);  friend void operator-=(PS& A, const PS& B);  friend PS operator-(const PS& A, const PS& B);  friend void operator\*=(PS& A, const PS& B);  friend PS operator\*(const PS& A, const PS& B);  friend void operator/=(PS& A, const PS& B);  friend PS operator/(const PS& A, const PS& B);  friend ostream& operator<<(ostream& os, PS& A);  friend istream& operator>>(istream& is, PS& A);  };  PS::PS() {  this->\_tu = this->\_mau = 0;  }  PS::PS(int a, int b) {  this->\_tu = a;  this->\_mau = b;  Handle();  }  PS::~PS() { ; }  void PS::set(int a, int b) {  this->\_tu = a;  this->\_mau = b;  Handle();  }  void PS::input() {  cin >> this->\_tu >> this->\_mau;  Handle();  }  void operator+=(PS& A, const PS& B) {  A.\_mau \*= B.\_mau;  A.\_tu \*= B.\_mau;  int value = B.\_tu \* A.\_mau;  A.\_tu += value;  A.Handle();  }  PS operator+(const PS& A, const PS& B) {  PS res = A;  res += B;  return res;  }  void operator-=(PS& A, const PS& B) {  A.\_mau \*= B.\_mau;  A.\_tu \*= B.\_mau;  int value = B.\_mau \* A.\_mau;  A.\_tu -= value;  A.Handle();  }  PS operator-(const PS& A, const PS& B) {  PS res = A;  res -= B;  res.Handle();  return res;  }  void operator\*=(PS& A, const PS& B) {  A.\_mau \*= B.\_mau;  A.\_tu \*= B.\_tu;  A.Handle();  }  PS operator\*(const PS& A, const PS& B) {  PS res = A;  res \*= B;  res.Handle();  return res;  }  void operator/=(PS& A, const PS& B) {  PS temp(B.\_mau, B.\_tu);  A \*= temp;  A.Handle();  }  PS operator/(const PS& A, const PS& B) {  PS res = A;  res /= B;  res.Handle();  return res;  }  ostream& operator<<(ostream& os, PS& A) {  if (A.\_mau == 0) {  os << "NaN";  return os;  }  os << A.\_tu << "/" << A.\_mau;  return os;  }  istream& operator>>(istream& is, PS& A) {  is >> A.\_tu >> A.\_mau;  return is;  }  PS Solve(int\* arr, int n) {  PS ps1;  ps1.set(arr[n - 1], 1);  for (int i = n - 2; i >= 0; i--) {  ps1 = PS(1, 1) / ps1;  PS temp(arr[i], 1);  ps1 += temp;  }  return ps1;  }  int main() {  int n;  cin >> n;  int\* arr = new int[n];  for (int i = 0; i < n; i++) cin >> arr[i];    PS res = Solve(arr, n);  cout << res;  return 0;  } |

**Bài 16.**Xây dựng lớp mẫu số phức thực hiện nhập vào n số phức tính tổng, hiệu, tích bằng phương thức toán tử

|  |
| --- |
| #include <iostream>  #include <math.h>  #include <limits.h>  using namespace std;  typedef long long ll;  class SP {  private :  double a, b;  // a + bi  public:  SP();  ~SP();  SP(double value, double value2);    // Operator  friend void operator+=(SP& A, const SP& B);  friend SP operator+(const SP& A, const SP& B);  friend void operator-=(SP& A, const SP& B);  friend SP operator-(const SP& A, const SP& B);  friend void operator\*=(SP& A, const SP& B);  friend SP operator\*(const SP& A, const SP& B);  friend void operator/=(SP& A, const SP& B);  friend SP operator/(const SP& A, const SP& B);  friend istream& operator>>(istream& is,SP& A);  friend ostream& operator<<(ostream& os,SP& A);  };  SP::SP() {  a = b = 0;  }  SP::SP(double value1, double value2) {  this->a = value1;  this->b = value2;  }  SP::~SP() { ; }  // OPERATOR  void operator+=(SP& A, const SP& B) {  A.a += B.a;  A.b += B.b;  }  SP operator+(const SP& A, const SP& B) {  SP res = A;  res += B;  return res;  }  void operator-=(SP& A, const SP& B) {  A.a -= B.a;  A.b -= B.b;  }  SP operator-(const SP& A, const SP& B) {  SP res = A;  res -= B;  return res;  }  void operator\*=(SP& A, const SP& B) {  A.a = (A.a \* B.a - A.b \* B.b);  A.b = (A.a \* B.b - A.b \* B.a);  }  SP operator\*(const SP& A, const SP& B) {  SP res = A;  res \*= B;  return res;  }  void operator/=(SP& A, const SP& B) {  A.a = (A.a \* B.a + A.b \* B.b) / (B.a \* B.a + B.b \* B.b);  A.b = (A.b \* B.a - A.a \* B.b) / (B.a \* B.a + B.b \* B.b);  }  SP operator/(const SP& A, const SP& B) {  SP res = A;  res /= B;  return res;  }  istream& operator>>(istream& is, SP& A) {  is >> A.a >> A.b;  return is;  }  ostream& operator<<(ostream& os, SP& A) {  os << A.a << " + " << A.b << "i";  return os;  }  int main()  {  SP num1, num2;  cout << "\nNhap 2 so phuc : \n";  cout << "SP1 = ";  cin >> num1;  cout << "SP2 = ";  cin >> num2;  SP res = num1 + num2;  cout << "\nTong : " << res << endl;  res = num1 - num2;  cout << "\nHieu : " << res << endl;  res = num1 \* num2;  cout << "\nTich : " << res << endl;  res = num1 / num2;  cout << "\nThuong : " << res << endl;  return 0;  } |

**Bài 17.**Xây dựng lớp biểu diễn các vector trong không gian n chiều có các phương thức toán tử: +, - hai vector, \* tích vô hướng hai véc tơ,-(đổi dấu) >>, <<.

**Bài 18.**Xây dựng lớp biểu diễn các đa thức với các phương thức toán tử: +, -, \* hai đa thức, tính giá trị đa thức, >>, <<.

|  |
| --- |
| #include <iostream>  #include <math.h>  #define MAX\_LEN 100  using namespace std;  class DT {  private:  int n; // level  float\* arr; // he so  public:  DT();  DT(int len);  ~DT();  double Fx(double x); // Tinh gia tri DT tai x  // Operator  friend istream& operator>>(istream& is, DT& A);  friend ostream& operator<<(ostream& os, const DT& A);    friend void operator+=(DT& A, const DT& B);  friend DT operator+(const DT& A, const DT& B);  friend void operator-=(DT& A, const DT& B);  friend DT operator-(const DT& A, const DT& B);  friend void operator\*=(DT& A, const DT& B);  friend DT operator\*(const DT& A, const DT& B);  friend void nhanDonThuc(const DT& A, int n, double value, int m, DT& B);  };  DT::DT() { ; }  DT::DT(int len) {  this->n = len;  arr = new float[len + 1] {};  }  DT::~DT() { ; }  double DT::Fx(double x) {  int curr = this->n;  double res = 0;  for (int i = 0; i <= n; i++) {  res += this->arr[i] \* pow(x, curr--);  }  return res;  }  istream& operator>>(istream& is, DT& A) {  cout << "Nhap so luong phan tu : ";  is >> A.n;  A.arr = new float[A.n + 1];  cout << "Nhap he so : \n";  for (int i = 0; i <= A.n; i++) {  is >> A.arr[i];  }  return is;  }  ostream& operator<<(ostream& os, const DT& A) {  int curr = A.n;  for (int i = 0; i <= A.n; i++) {  if (i != A.n)  os << A.arr[i] << "x" << "^" << curr-- << " + ";  else  os << A.arr[i];  }  return os;  }  void operator+=(DT& A, const DT& B) {  int i, j, k, len;  DT res(max(A.n, B.n));  i = j = k = 0;  if (A.n > B.n) {  len = A.n - B.n;  while (i < len) {  res.arr[k++] = A.arr[i++];  }  }  else if (A.n < B.n) {  len = B.n - A.n;  while (j < len) {  res.arr[k++] = B.arr[j++];  }  }  for (; i <= A.n && j <= B.n; i++, j++, k++) {  res.arr[k] = A.arr[i] + B.arr[j];  }  A = res;  }  DT operator+(const DT& A, const DT& B) {  DT res = A;  res += B;  return res;  }  void operator-=(DT& A, const DT& B) {  int i, j, k, len;  DT res(max(A.n, B.n));  i = j = k = 0;  if (A.n > B.n) {  len = A.n - B.n;  while (i < len) {  res.arr[k++] = A.arr[i++];  }  }  else if (A.n < B.n) {  len = B.n - A.n;  while (j < len) {  res.arr[k++] = -B.arr[j++];  }  }  for (; i <= A.n && j <= B.n; i++, j++, k++) {  res.arr[k] = A.arr[i] - B.arr[j];  }  A = res;  }  DT operator-(const DT& A, const DT& B) {  DT res = A;  res -= B;  return res;  }  void nhanDonThuc(const DT& A, int n, double value, int m, DT& B) {  B.n = n + m;  for (int i = 0; i < m; i++) {  B.arr[i] = 0;  }  for (int i = m; i <= n + m; i++) {  B.arr[i] = value \* A.arr[i - m];  }  }  void operator\*=(DT& A, const DT& B) {  DT res;  DT temp(MAX\_LEN);  int k;  nhanDonThuc(B, B.n, A.arr[0], 0, res);  for (int i = 1; i <= A.n; i++) {  nhanDonThuc(B, B.n, A.arr[i], i, temp);  res += temp;  }  A = res;  }  DT operator\*(const DT& A, const DT& B) {  DT res = A;  res \*= B;  return res;  }  int main()  {  DT dt1, dt2, res;  cin >> dt1 >> dt2;  res = dt1 \* dt2;  cout << res;  return 0;  } |

**Bài 19.**Xây dựng lớp biểu diễn các ma trận có các phương thức toán tử: +, - ,\* hai ma trận, >>, <<.

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  class matrix {  private:  int \_row, \_column;  double\*\* a;  public:  matrix();  matrix(int r,int cl);  matrix(const matrix& p);  ~matrix();    int Row();  int Col();  // Operator  friend istream& operator>>(istream& is, matrix& A);  friend ostream& operator<<(ostream& os, matrix& A);  friend void operator+=(matrix& A, const matrix& B);  friend matrix operator+(const matrix& A, const matrix& B);  friend void operator-=(matrix& A, const matrix& B);  friend matrix operator-(const matrix& A, const matrix& B);  friend matrix operator\*(const matrix& A, const matrix& B);  };  matrix::matrix() {  a = NULL;  this->\_column = this->\_row = 0;  }  matrix::matrix(int r, int cl) {  this->a = new double\* [r];  for (int i = 0; i < r; i++) {  this->a[i] = new double[r] {};  }  this->\_row = r;  this->\_column = cl;  }  matrix::matrix(const matrix& p) {  \*this = p;  }  matrix::~matrix() { ; }  int matrix::Row() {  return this->\_row;  }  int matrix::Col() {  return this->\_column;  }  istream& operator>>(istream& is, matrix& A) {  cout << "Nhap so hang : ";  is >> A.\_row;  cout << "Nhap so cot : ";  is >> A.\_column;  A.a = new double\* [A.\_row]{};  for (int i = 0; i < A.\_row; i++) {  A.a[i] = new double[A.\_column];  }  cout << "Nhap ma tran : \n";  for (int i = 0; i < A.\_row; i++) {  for (int j = 0; j < A.\_column; j++) {  is >> A.a[i][j];  }  }  return is;  }  ostream& operator<<(ostream & os,matrix& A) {  for (int i = 0; i < A.\_row; i++) {  for (int j = 0; j < A.\_column; j++) {  os << A.a[i][j] << " ";  }  os << endl;  }  return os;  }  void operator+=(matrix& A, const matrix& B) {  if (A.\_row != B.\_row || A.\_column != B.\_column) {  return;  }  for (int i = 0; i < A.\_row; i++) {  for (int j = 0; j < A.\_column; j++) {  A.a[i][j] += B.a[i][j];  }  }  }  matrix operator+(const matrix& A, const matrix& B) {  matrix res = A;  res += B;  return res;  }  void operator-=(matrix& A, const matrix& B) {  if (A.\_row != B.\_row || A.\_column != B.\_column) {  return;  }  for (int i = 0; i < A.\_row; i++) {  for (int j = 0; j < A.\_column; j++) {  A.a[i][j] -= B.a[i][j];  }  }  }  matrix operator-(const matrix& A, const matrix& B) {  matrix res = A;  res -= B;  return res;  }  matrix operator\*(const matrix& A, const matrix& B) {  matrix res(A.\_row, B.\_column);  if (A.\_column == B.\_row) {  for (int i = 0; i < A.\_row; i++) {  for (int j = 0; j < B.\_column; j++) {  for (int k = 0; k < A.\_column; k++) {  res.a[i][j] += (A.a[i][k] \* B.a[k][j]);  }  }  }  }  return res;  }  int main()  {  matrix A, B, res;  cin >> A;  cin >> B;  cout << "\nMa tran vua nhap : \n";  cout << A << endl;  cout << B << endl;    cout << "\n1. Tong" << endl;  cout << "\n2. Hieu" << endl;  cout << "\n3. Tich" << endl;  int c;  cin >> c;  switch (c)  {  case 1:  if (A.Col() != B.Col() || A.Row() != B.Row())  cout << "\nKhong the tinh tong" << endl;  else {  res = A + B;  cout << res;  }  break;  case 2:  if (A.Col() != B.Col() || A.Row() != B.Row())  cout << "\nKhong the tinh hieu" << endl;  else {  res = A - B;  cout << res;  }  break;  case 3:  if (A.Col() != B.Row())  cout << "\nKhong the tinh tich" << endl;  else {  res = A \* B;  cout << res;  }  break;  default:  break;  }  return 0;  } |

1. **Phân tích độ phức tạp**

**Bài 20.** Cho hàm sau phân tích độ phức tạp về thời gian

|  |
| --- |
| **int** sum**(int** n**)**  **{**  **int** s**=**1**; // 1**  **for(int** i**=**n**;**i**>**0**;**i**-=**3**)** // n/3 + 3  s**\*=**i**; // 2n/3**  **return** s**;** // 1  **}** |
| **= 1 + n/3 + 3 + 2n/3 + 1 = n + 5** |

**Bài 21.** Cho hàm sau phân tích độ phức tạp về thời gian

|  |
| --- |
| **void** sort**(int** n**,int\***a**,bool** ok**=true)**  **{**  **for(int** i**=**0**;**i**<**n**;**i**++) // 3 + n**  **for(int** j**=**i**+**1**;**j**<**n**;**j**++) // (n-1)+(n-2)+…+1+3**  **if(**a**[**i**]>**a**[**j**]==**ok**) // 2**  **{**  **int** t**=**a**[**i**]; // 1**  a**[**i**]=**a**[**j**]; // 1**  a**[**j**]=**t**; // 1**  **}**  **}** |

**Bài 22.** Cho hàm sau phân tích độ phức tạp về thời gian

|  |
| --- |
| **int** fibo**(int** n**)**  **{**  **int** f**[**n**+**5**];**  f**[**0**]=**f**[**1**]=**1**;**  **for(int** i**=**2**;**i**<**n**;**i**++)** f**[**i**]=**f**[**i**-**1**]+**f**[**i**-**2**];**  **return** f**[**n**];**  **}** |

**Bài 23.** Cho hàm sau phân tích độ phức tạp về thời gian

|  |
| --- |
| **int** TimKiemNhiPhan**(int** n**,int\***a**,int** x**){**  **int** L **=**1**,** R **=**n **;**  **while(**L**<=**R**)**  **{**  **int** M **=(**L **+** R**)/**2**;**  **if(**x**==**a**[**M**])return** M**;**  **if(**x **<** a**[**M**])** R**=** M **-**1**;**  **else** L **=** M **+**1**;**  **}**  **return**0**;**  **}** |

**Bài 23.** Cho hàm sau phân tích độ phức tạp về thời gian

|  |
| --- |
| **bool** isprime**(int** n**)**  **{**  **if(**n**==**2**)returntrue;**  **if(**n**<**2**||(**n**%**2**==**0**))returnfalse;**  **for(int** i**=**3**;**i**\***i**<=**n**;**i**+=**2**)**  **if(**n**%**i**==**0**)returnfalse;**  **returntrue;**  **}** |

1. **Đệ quy**

**Bài 24.** Viết hàm đệ quy tính n! và hàm main mô phỏng

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  typedef long long ll;  ll F[100]{};  ll giaiThua(ll n) {  if (n <= 1) {  F[n] = 1;  return 1;  }  if (F[n])  return F[n];  F[n] = n \* giaiThua(n - 1);  return F[n];  }  int main()  {  ll n;  cin >> n;  for (int i = 2; i <= n; i++)  giaiThua(i);  cout << n << "! = " << F[n] << endl;  return 0;  } |

**Bài 25.** Viết hàm đệ quy tính tổ hợp chập k của n và hàm main mô phỏng

|  |
| --- |
| #include <iostream>  using namespace std;  typedef long long ll;  ll F[200]{};  ll factorial(int n) {  if (F[n])  return F[n];  if (n <= 1)  {  F[n] = 1;  return 1;  }  else {  F[n] = n \* factorial(n - 1);  return F[n];  }  }  ll nCk(int n, int k) {  return factorial(n) / (factorial(k) \* factorial(n - k));  }  int main() {  int n, k;  cin >> n >> k;  for (int i = 1; i <= n; i++) {  factorial(i);  }  cout << nCk(n, k);  return 0;  } |

**Bài 26.** Viết hàm đệ quy in ra màn hình tam giác \*

|  |
| --- |
| #include <iostream>  using namespace std;  void PrintTriangle(int n) {  if (n == 0) return;  else {  PrintTriangle(n - 1);  for (int i = 0; i < n; i++) cout << "\*";  cout << endl;  }  }  int main()  {  int n;  cin >> n;  PrintTriangle(n);  return 0;  } |

**Bài 27.** Viết hàm đệ quy tính tổng của dãy a1+a2+...+an

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  int SumArr(int\* arr, int n) {  if (n == 1)  return arr[0];  else  return arr[n - 1] + SumArr(arr, n - 1);  }  int main()  {  int n;  cin >> n;  int\* arr = new int[n] {};  for (int i = 0; i < n; i++) cin >> arr[i];  cout << SumArr(arr, n);  return 0;  } |

**Bài 28.** Viết hàm đệ quy tính số fibonacci có sử dụng một mảng phụ trợ nếu đã tính rồi thì tra trong mảng nếu chưa có mới tính

|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  long long F[200]{};  long long fib(int n) {  if (F[n] > 0)  return F[n];  else {  long long res = (fib(n - 1) + fib(n - 2));  F[n] = res;  return F[n];  }  }  int main()  {  F[0] = F[1] = 1;  int n;  cin >> n;  cout << "fib(n) = " << fib(n);  return 0;  } |

**Bài 29.** Viết hàm đệ quy tương hỗ tính sinx và cosx với sai số epsilon theo công thức





|  |
| --- |
| #include <iostream>  #include <math.h>  using namespace std;  double epsilon;  double Cos(double x);  double Sin(double x) {  if (x > -epsilon && x < epsilon) {  return x;  }  else {  return 2 \* Sin(x / 2) \* Cos(x / 2);  }  }  double Cos(double x) {  if (x > -epsilon && x < epsilon) {  return 1;  }  else {  return pow(Cos(x / 2), 2) - pow(Sin(x / 2), 2);  }  }  int main()  {  double x;  cout << "Nhap epsilon : ";  cin >> epsilon;  cout << "Nhap x : ";  cin >> x;  cout << "sin(" << x << ") = " << Sin(x) << endl;  cout << "cos(" << x << ") = " << Cos(x) << endl;  return 0;  } |

**Bài 30.** Viết hàm đệ quy mô phỏng chuyển Tháp Hà Nội

|  |
| --- |
| #include <iostream>  using namespace std;  // n : Dia thu n  // T1, T2, T3 : ten cot  void HN\_tower(int n, char T1, char T2, char T3)  {  if (n == 0) {  return;  }  HN\_tower(n - 1, T1, T3, T2);  cout << "Chuyen dia " << n << " tu " << T1 << " sang " << T2 << endl;  HN\_tower(n - 1, T3, T2, T1);  }  int main() {  int n;  cout << "Nhap so luong dia : ";  cin >> n;  HN\_tower(n, 'A', 'B', 'C');  return 0;  } |

1. **Cấu trúc dữ liệu tuyến tính**

**Bài 31.** Áp dụng vector trong thư viện STL xây dựng lớp đa thức thực hiện tính tổng hai đa thức.

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  void displayDT(vector<int> arr) {  int curr = arr.size() - 1;  for (int i = 0; i < arr.size(); i++) {  if (curr == 0 && arr[i] != 0) cout << arr[i];  else  cout << arr[i] << "x^" << curr << " + ";  curr--;  }  }  long long Sum2DT(vector<int> arr1, vector<int> arr2, int x) {  while (arr1.size() < arr2.size()) arr1.insert(arr1.begin(), 0);  while (arr1.size() > arr2.size()) arr1.insert(arr2.begin(), 0);  long long res = 0;  int curr = arr1.size() - 1;  for (int i = 0; i < arr1.size(); i++) {  res += arr1[i] + arr2[i];  res \*= pow(x, curr--);  }  return res;  }  int main()  {  int n, m;  cin >> n >> m;  vector<int> arr1(n + 1);  vector<int> arr2(m + 1);  for (int i = 0; i <= n; i++) cin >> arr1[i];  for (int i = 0; i <= m; i++) cin >> arr2[i];  displayDT(arr1);  cout << endl;  displayDT(arr2);  cout << endl;  int value;  cin >> value;  cout << "Fx(" << value << ") + Gx(" << value << ") = " << Sum2DT(arr1, arr2, value);  return 0;  } |

**Bài 32.** Áp dụng vector trong thư viện STL xây dựng lớp số lớn để giải bài toán tính giai thừa tới tối đa 1000!

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  vector<int> multiplyLargeNumber(vector<int> num1, int x) {  int surplus = 0;  for (int i = 0; i < num1.size(); i++) {  int temp = surplus + num1[i] \* x;  num1[i] = temp % 10;  surplus = temp / 10;  }  while (surplus != 0) {  num1.push\_back(surplus % 10);  surplus /= 10;  }  return num1;  }  int main() {  vector<int> num1(1, 2);  int n;  cout << "Nhap so n : ";  cin >> n;  cout << n << "! = ";  for (int i = 3; i <= n; i++) {  num1 = multiplyLargeNumber(num1, i);  }  for (int i = num1.size() - 1; i >= 0; i--) cout << num1[i];  return 0;  } |

**Bài 33**. Áp dụngvector trong thư viện STL xây dựng lớp lớp học gồm n sinh viên và các sinh viên có thông tin về tên, tuổi, điểm. Nhập danh sách n sinh viên tính điểm trung bình, sắp xếp danh sách theo điểm giảm dần.

|  |
| --- |
| #include <iostream>  #include <vector>  #include <stdlib.h>  #include <string>  using namespace std;  class student {  private:  string \_fullName;  int \_age;  string \_idStudent;  string \_address;  float \_mark;  public:  student();  ~student();  void input();  void display();  friend void sort\_Student\_List(vector<student>& arr);  friend void swapStudent(student& A, student& B);  };  student::student() { ; }  student::~student() { ; }  string inSTRING() {  string res;  do {  getline(cin, res);  } while (res == "");  return res;  }  void student::input() {  cout << "\tTen : ";  this->\_fullName = inSTRING();  cout << "\tMa sv : ";  this->\_idStudent = inSTRING();  cout << "\tTuoi : ";  cin >> this->\_age;  cout << "\tDia chi : ";  this->\_address = inSTRING();  cout << "\tDiem : ";  cin >> this->\_mark;  }  void student::display() {  cout << "\t" << this->\_fullName;  cout << "\t" << this->\_age;  cout << "\t" << this->\_idStudent;  cout << "\t" << this->\_address;  cout << "\t" << this->\_mark;  cout << endl;  }  void swapStudent(student& A, student& B) {  student tempST = A;  A = B;  B = tempST;  }  void sort\_Student\_List(vector<student>& arr) {  for (int i = 0; i < arr.size() - 1; i++) {  for (int j = i + 1; j < arr.size(); j++) {  if (arr[i].\_mark < arr[j].\_mark)  swapStudent(arr[i], arr[j]);  }  }  }  int main() {  int n;  cin >> n;  vector<student> arr(n);  for (int i = 0; i < n; i++) {  cout << "Nhap SV[" << i << "] : " << endl;  arr[i].input();  }  cout << "==================================================================================" << endl;  cout << "Danh sach sv vua nhap : " << endl;  for (int i = 0; i < n; i++) {  cout << i + 1;  arr[i].display();  }  cout << "-----------------------------------------------------------------------------------" << endl;  // push a notification  cout << "SORT !" << endl;  sort\_Student\_List(arr);  //  cout << "==================================================================================" << endl;  cout << "Danh sach sv vua nhap : " << endl;  for (int i = 0; i < n; i++) {  cout << i + 1;  arr[i].display();  }  cout << "-----------------------------------------------------------------------------------" << endl;  return 0;  } |

**Bài 34.** Áp dụngvector trong thư viện STL xây dựng lớp ma trận tìm ma trận chuyển vị

|  |
| --- |
| #include <iostream>  #include <vector>  using namespace std;  class Matrix {  private:  vector<vector<int>> T;  int \_row, \_column;  public:  Matrix();  Matrix(int r, int cl);  ~Matrix();  void input();  void display();  Matrix CvMatrix();  };  Matrix::Matrix() { ; }  Matrix::~Matrix() { ; }  Matrix::Matrix(int r, int cl) {  T.resize(r, vector<int>(cl));  this->\_row = r;  this->\_column = cl;  }  void Matrix::input() {  cin >> \_row >> \_column;  T.resize(\_row, vector<int>(\_column));  for (int i = 0; i < \_row; i++) {  for (int j = 0; j < \_column; j++) {  cin >> T[i][j];  }  }  }  void Matrix::display() {  for (int i = 0; i < \_row; i++) {  for (int j = 0; j < \_column; j++) {  cout << T[i][j] << " ";  }  cout << endl;  }  }  Matrix Matrix::CvMatrix() {  Matrix res(\_column, \_row);  for (int i = 0; i < \_row; i++) {  for (int j = 0; j < \_column; j++) {  res.T[j][i] = this->T[i][j];  }  }    return res;  }  int main()  {  Matrix m1;  Matrix m2;  m1.input();  m1.display();  m2 = m1.CvMatrix();  m2.display();  return 0;  } |

**Bài 34.** Áp dụngvector trong thư viện STL xây dựng lớp tập hợp cho phép rút gọn tập hợp, tìm hợp, giao của hai tập hợp.

|  |
| --- |
| #include <iostream>  #include <vector>  #include <string>  using namespace std;  template<typename T>  class collection {  private:  vector<T> arr;  public:  collection();  collection(int n);  ~collection();  void push(T element);  void input();  void display();  // Find Common elements  template<typename T> friend collection<T> Find\_common\_element(collection<T> A, collection<T> B);  };  template<typename T> collection<T>::collection() {  ;  }  template<typename T> collection<T>::collection(int n) {  arr.resize(n);  }  template<typename T> collection<T>::~collection() {  ;  }  template<typename T> void collection<T>::push(T element) {  for (int i = 0; i < this->arr.size(); i++) {  if (this->arr[i] == element)  return;  }    this->arr.push\_back(element);  }  template<typename T> void collection<T>::input() {  int n = 0;  if (!this->arr.size())  {  cout << "\nNhap so luong : ";  cin >> n;  arr.resize(n);  }  cout << "\nNhap phan tu trong tap hop : (" << n << " phan tu) :\n";  for (int i = 0; i < n; i++)  {  cin >> arr[i];  }  }  template<typename T> void collection<T>::display() {  cout << "\n{";  for (int i = 0; i < this->arr.size(); i++) {  if (i) cout << "; ";  cout << this->arr[i];  }  cout << "}\n";  }  template<typename T> collection<T> Find\_common\_element(collection<T> A, collection<T> B) {  collection<T> res(0);  for (auto& x : A.arr) {  for (int i = 0; i < B.arr.size(); i++) {  if (B.arr[i] == x)  res.push(x);  }  }  for (auto& x : B.arr) {  for (int i = 0; i < A.arr.size(); i++) {  if (A.arr[i] == x)  res.push(x);  }  }  return res;  }  template<typename T> void Solution(T type) {  cout << "\nChuong trinh tim giao 2 tap hop : \n";    collection<T> arr1;  collection<T> arr2;  arr1.input();  arr2.input();  cout << "\nTap hop vua nhap : \n";  arr1.display();  arr2.display();  cout << "\nGiao cua 2 tap hop : \n";  collection<T> res = Find\_common\_element(arr1, arr2);  res.display();  }  int main() {  int choose;  cout << "Chon loai tap hop : ";  cout << "\n1. chuoi ky tu";  cout << "\n2. so nguyen";  cout << "\n3. so thuc";  cin >> choose;  switch (choose)  {  case 1:  Solution(string());  break;  case 2:  Solution(int());  break;  case 3:  Solution(double());  break;  default:  break;  }  return 0;  } |

**Bài 35.** Cài đặt cấu trúc vector trừu tượng bổ sung thêm phương thức sắp xếp tăng dần các phần tử, phương thức tìm max, min.

|  |
| --- |
| #include <iostream>  using namespace std;  template<typename T> struct Node {  T \_data;  Node<T>\* \_next;  };  template<typename T> class VECTOR {  private:  Node<T>\* root;  Node<T>\* createNode(T value);  int len;  Node<T>\*\* get(int id);  public:  VECTOR();  ~VECTOR();  int size();  void push(T value);  void erase(int k);  void pop();  void display();  friend void swap(Node<T>\*\* a, Node<T>\*\* b);  void BBsort();  T MAX();  T MIN();  int find(T value);    };  template<typename T> VECTOR<T>::VECTOR() {  root = NULL;  }  template<typename T> VECTOR<T>::~VECTOR() {  delete root;  }  template<typename T> Node<T>\* VECTOR<T>::createNode(T value) {  Node<T>\* p = new Node<T>();  p->\_data = value;  p->\_next = NULL;  return p;  }  template<typename T> Node<T>\*\* VECTOR<T>::get(int id) {  if (root == NULL) return NULL;  if (id == 0) {  Node<T>\*\* res = &root;  return res;  }  Node<T>\* curr = root;  Node<T>\* p = root;  while (id > 0 && curr->\_next != NULL)  {  p = curr;  curr = curr->\_next;  id--;  }  Node<T>\*\* res = NULL;  if (id == 0) {  res = &(p->\_next);  return res;  }  else  return NULL;  }  template<typename T> int VECTOR<T>::size() {  return this->len;  }  template<typename T> void VECTOR<T>::push(T value) {  if (root == NULL) {  root = createNode(value);  len++;  return;  }  else {  Node<T>\* curr = root;  while (curr->\_next != NULL) curr = curr->\_next;  curr->\_next = createNode(value);  len++;  }  }  template<typename T> void VECTOR<T>::pop() {  if (root != NULL) {  Node<T>\* curr = root;  Node<T>\* p = curr;  while (curr->\_next != NULL) {  p = curr;  curr = curr->\_next;  }  delete p->\_next;  p->\_next = NULL;  len--;  }  }  template<typename T> void VECTOR<T>::erase(int k) {  // Erase Kth number  if (this->root == NULL)  return;  k--;  Node<T>\* curr = this->root;  Node<T>\* p = curr;  while (k > 0 && curr->\_next != NULL) {  p = curr;  curr = curr->\_next;  k--;  }  if (k == 0) {  p->\_next = curr->\_next;  len--;  }  }  template<typename T> void VECTOR<T>::display() {  if (root == NULL) return;    Node<T>\* curr = root;  while (curr != NULL) {  cout << curr->\_data << " ";  curr = curr->\_next;  }  cout << endl;  }  template<typename T> int VECTOR<T>::find(T value) {  // The func return the first position, have value in list  if (root == NULL) return -1;  Node<T>\* curr = root;  int k = 0;  while (curr->\_next != NULL && curr->\_data != value) {  curr = curr->\_next;  k++;  }  if (curr->\_data == value)  return k;  else  return -1;  }  template<typename T> void swap(Node<T>\*\* a, Node<T>\*\* b) {  T temp = (\*a)->\_data;  (\*a)->\_data = (\*b)->\_data;  (\*b)->\_data = temp;  }  template<typename T> void VECTOR<T>::BBsort() {  Node<T>\*\* v1;  Node<T>\*\* v2;  v1 = NULL;  v2 = NULL;  for (int i = 0; i < len - 1; i++) {  for (int j = i + 1; j < len; j++) {  v1 = this->get(i);  v2 = this->get(j);  if (v1 != NULL && v2 != NULL  && (\*v1)->\_data > (\*v2)->\_data)  {  swap(v1, v2);  }  }  }  }  template<typename T> T VECTOR<T>::MAX() {  if (root == NULL)  return T();  Node<T>\* curr = root;  T res = curr->\_data;  while (curr != NULL) {  if (curr->\_data > res) {  res = curr->\_data;  }  curr = curr->\_next;  }  return res;  }  template<typename T> T VECTOR<T>::MIN() {  if (root == NULL)  return T();  Node<T>\* curr = root;  T res = curr->\_data;  while (curr != NULL) {  if (curr->\_data < res) {  res = curr->\_data;  }  curr = curr->\_next;  }  return res;  }  int main() {  VECTOR<int> arr;  arr.push(2);  arr.push(1);  arr.push(4);  arr.push(3);  arr.display();  cout << "\nBuble Sort !!\n";  arr.BBsort();  arr.display();  cout << "Max = " << arr.MAX() << endl;  cout << "Min = " << arr.MIN() << endl;  return 0;  } |

1. **Cấu trúc dữ liệu phi tuyến**

|  |
| --- |
|  |