Đề mẫu:

1. Tính số phép toán cơ bản nhiều nhất mà thuật toán dưới đây cần phải thực hiện. Xác định độ phức tạp tiệm cận của thuật toán.

|  |  |
| --- | --- |
| int Count2(int \*a,int n) {  int k=0;  for(int i=1;i<=n;i++)  for(int j=i+1;j<=n;j++)  if((a[i]+a[j])%2==0  && a[i]>a[j])  k++;  return k;  } | (1)  (n + 3)  3 + (n-1) + (n-2) + … + 1  5 \* [(n-1) + (n-2) + … + 1]  (1) |
| Tổng phép toán = 1 + n + 3 + 3 + (n-1) + (n-2) + … + 1  + 5 \* [(n-1) + (n-2) + … + 1] + 1 = 3n2 – 2n + 9  => Độ phức tạp thời gian: O(n2) | |

1. Cài đặt class node và slist:

|  |  |
| --- | --- |
| #include <iostream>  using namespace std;  template<class T>  class node {  private:  node\* next;  T elem;  public:  node() {  this->next = NULL;  this->elem = T();  }  node(T value, node\* p) {  this->elem = value;  this->next = p;  }  node\* getNext() {  return this->next;  }  T& getElem() {  return this->elem;  }  void setElem(T value) {  this->elem = value;  }  void setNext(node\* p) {  this->next = p;  }  }; | template<class T>  class slist {  private:  int num;  node<T>\* head, \* trail;  public:  slist() {  num = 0;  head = trail = NULL;  }  void push\_front(T elem) {  if (head == NULL && trail == NULL) {  head = trail = new node<T>(elem, NULL);  return;  }  node<T>\* p = new node<T>(elem, head);  head = p;  }  void erase(T elem) {  if (head == NULL) return;  if (head->getElem() == elem) {  head = head->getNext();  }  node<T>\* curr = head;  while (curr->getNext() != NULL)  {  if (curr->getNext()->getElem() == elem) {  curr->setNext(curr->getNext()->getNext());  }  else  curr = curr->getNext();  }  }  T& begin() {  return this->head->getElem();  }  T& end() {  return this->trail->getElem();  }  void display() {  node<T>\* curr = head;  while (curr != NULL)  {  cout << curr->getElem() << " ";  curr = curr->getNext();  }  }  };  int main() {  slist<int> list;  list.push\_front(1);  list.push\_front(2);  list.push\_front(5);  list.push\_front(4);  list.push\_front(5);  list.erase(5);  list.display();  cout << "\nBegin: " << list.begin() << endl;  cout << "End: " << list.end() << endl;  return 0;  } |

1. (3.0 điểm) Thực hiện các yêu cầu sau đây: - Xây dựng lớp biểu diễn các đối tượng Tivi gồm các thuộc tính: mã hàng (không trùng nhau),Hãng SX, tên Tivi, đơn giá, số lượng; Các toán tử nhập >>, xuất <<

|  |
| --- |
| #include <iostream>  #include <string>  #include <string.h>  #include <vector>  #include <algorithm>  using namespace std;  class TV {  private:  int id;  string hangSx;  string name;  int cost;  int num;  public:  TV() {  id = cost = num = 0;  hangSx = name = "";  }  TV(int id, string hangSx, string name, int cost, int num) {  this->id = id;  this->hangSx = hangSx;  this->name = name;  this->cost = cost;  this->num = num;  }  friend istream& operator>>(istream& is, TV& A) {  cout << "\nNhap thong tin TV: " << endl;  cout << "Ma Tv: "; is >> A.id;  cout << "Hang san xuat: ";  is.ignore(); getline(is, A.hangSx);  cout << "Ten: ";  is.ignore(); getline(is, A.name);  cout << "Don gia: ";  is >> A.cost;  cout << "So luong: "; is >> A.num;  return is;  }  friend ostream& operator<<(ostream& os, TV A) {  os << "Thong tin TV(" << A.id << "): " << endl;  os << "Ma Tv: " << A.id << endl;  os << "Hang san xuat: " << A.hangSx << endl;  os << "Ten: " << A.name << endl;  os << "Don gia: " << A.cost << endl;  os << "So luong: " << A.num << endl;  return os;  }  string& gethangSx() {  return this->hangSx;  }  int& getID() {  return this->id;  }  int donGia() {  return this->num \* this->cost;  }  };  bool cmp(TV a, TV b) {  return a.getID() < b.getID();  }  class listTV {  private:  vector<TV> buff;  public:  listTV() {  ;  }  bool Add(TV A) {  int id = A.getID();  for (auto& x : buff) {  if (x.getID() == id) return false;  }  buff.push\_back(A);  return true;  }  bool Remove(int id) {  vector<TV>::iterator it = buff.begin();  for (it; it != buff.end(); it++) {  if ((\*it).getID() == id)  break;  }  if (it != buff.end()) {  buff.erase(it);  return true;  }  return false;  }  int totalCost(string hangSx) {  int total = 0;  for (auto& x : buff) {  if (x.gethangSx() == hangSx) {  total += x.donGia();  }  }  return total;  }  TV& find(int id) {  sort(buff.begin(), buff.end(), cmp);  int l = 0;  int r = buff.size() - 1;  while (l <= r) {  int mid = (l + r) / 2;  if (buff[mid].getID() == id) return buff[mid];  else {  if (buff[mid].getID() < id) l = mid + 1;  else  r = mid - 1;  }  }  TV p;  return p;  }  };  int main() {  listTV arr;  int n;  cout << "\nNhap so luong TV nhap vao: "; cin >> n;  for (int i = 0; i < n; i++) {  TV t;  cin >> t;  if (arr.Add(t));  else  {  cout << "Ma TV nay da trung ! Hay nhap lai\n";  i--;  }  }  int id;  cout << "Nhap ma TV can tim: "; cin >> id;  TV res = arr.find(id);  if (res.getID())  cout << res;  else  cout << "Khong tim thay !";  return 0;  } |