**I.Câu 17**

**I.1 Tên đề bài**Cho 3 số thực x, y, z. Viết chương trình sử dụng hàm hãy tìm

a. max(x,y,z) và min(x,y,z)

b. min(x+y+z, xyz)

c. min2(x+y+z/2, xyz)+1

**I.2 Thuật toán**Bước 1: Nhập dữ liệu

1. Yêu cầu người dùng nhập ba số nguyên lớn x, y, z.
2. Lưu các giá trị này để sử dụng trong các phép tính tiếp theo.

Bước 2: Tính giá trị lớn nhất max(x, y, z)

1. So sánh y và z:
   * Tìm giá trị lớn hơn giữa y và z, gọi kết quả này là max\_yz.
2. So sánh x và max\_yz:
   * Tìm giá trị lớn hơn giữa x và max\_yz. Đây là giá trị lớn nhất trong ba số x, y, z.

Bước 3: Tính giá trị nhỏ nhất min(x, y, z)

1. So sánh y và z:
   * Tìm giá trị nhỏ hơn giữa y và z, gọi kết quả này là min\_yz.
2. So sánh x và min\_yz:
   * Tìm giá trị nhỏ hơn giữa x và min\_yz. Đây là giá trị nhỏ nhất trong ba số x, y, z.

Bước 4: Tính giá trị nhỏ hơn giữa tổng và tích của x, y, z

1. Tính tổng của x, y, và z:
   * sum = x + y + z.
2. Tính tích của x, y, và z:
   * product = x \* y \* z.
3. So sánh sum và product:
   * Lấy giá trị nhỏ hơn giữa sum và product.

Bước 5: Tính giá trị biểu thức phức tạp

1. Tính tổng của x, y, và z/2:
   * half\_sum = x + y + z/2.
2. So sánh half\_sum với product:
   * Lấy giá trị nhỏ hơn giữa half\_sum và product.
3. Bình phương giá trị nhỏ hơn vừa tìm được.
4. Cộng thêm 1 vào kết quả bình phương.

Bước 6: Xuất kết quả

1. In ra giá trị lớn nhất max(x, y, z).
2. In ra giá trị nhỏ nhất min(x, y, z).
3. In ra giá trị nhỏ hơn giữa tổng và tích của x, y, z.
4. In ra kết quả của biểu thức phức tạp: (min(x + y + z/2, x \* y \* z))^2 + 1.

**I.3 Chương trình nguồn**#include <stdio.h>

#define ll long long

ll somax(ll a,ll b){

    if(a>b) return a; else return b;

}

ll somin(ll a,ll b){

    if(a>b) return b; else return a;

}

int main(){

    ll x,y,z;

    printf("Nhap x: "); scanf("%lld", &x);

    printf("Nhap y: "); scanf("%lld", &y);

    printf("Nhap z: "); scanf("%lld", &z);

    printf("max(x,y,z)= %lld\n", somax(x,somax(y,z)));

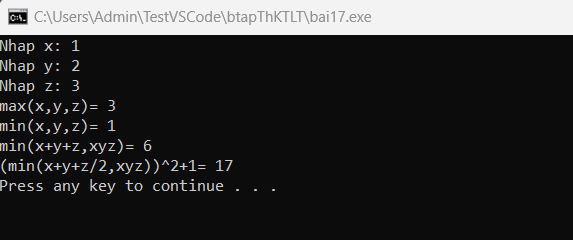
    printf("min(x,y,z)= %lld\n", somin(x,somin(y,z)));

    printf("min(x+y+z,xyz)= %lld\n", somin(x+y+z,x\*y\*z));

    printf("(min(x+y+z/2,xyz))^2+1= %lld\n", somin(x+y+z/2,x\*y\*z)\*somin(x+y+z/2,x\*y\*z)+1);

    return 0;

}

**I.4 Kết quả chương trình  
**

**II. Câu 26**

**II.1 Tên đề bài**a) Nhập vào số thực x và hãy viết chương trình sử dụng hàm tính giá trị của biểu thức sau:   
(x-2)(x-4)…(x-64)/(x-1)(x-3)…(x-63)  
b) Với số tự nhiên n cho trước. Hãy viết chương trình có sử dụng hàm để tính biểu thức sau: 1.2+2.3.4+…+ n(n+1)…(2n)

**II.2 Thuật toán**1. Hàm check(x)

* Chức năng: Kiểm tra xem giá trị x có bằng một trong các số lẻ từ 1 đến 63 không.
* Bước thực hiện:
  1. Duyệt qua tất cả các số lẻ từ 1 đến 63 bằng vòng lặp, với bước nhảy là 2.
  2. Nếu x bằng một số trong dãy này, trả về 1 (biểu thức không tồn tại).
  3. Nếu không có số nào thỏa mãn, kết thúc vòng lặp.

2. Hàm bt1(x)

* Chức năng: Tính giá trị biểu thức A
* Bước thực hiện:
  1. Khởi tạo giá trị gt = 1 (tích ban đầu).
  2. Duyệt qua tất cả các số lẻ từ 1 đến 63 bằng vòng lặp.
  3. Với mỗi số i, tính toán 1+(-1)/(x-i)
  4. Nhân giá trị này vào gt.
  5. Sau khi hoàn tất vòng lặp, trả về giá trị cuối cùng của gt.

3. Hàm bt2(n)

* Chức năng: Tính giá trị biểu thức B, trong đó j tăng từ 1 trở đi.
* Bước thực hiện:
  1. Khởi tạo giá trị:
     + t = 2 (giá trị khởi đầu của tổng).
     + temp = 2 (biểu thức ban đầu của tích).
     + i = 3 (chỉ số đầu tiên của dãy số lẻ từ 3).
     + j = 1 (chỉ số chia khởi đầu).
  2. Duyệt qua các số lẻ từ 3 đến 2n với bước nhảy là 2.
  3. Tại mỗi bước:
     + Cập nhật temp với công thức temp=temp×i×(i+1)/j
     + Cộng temp vào tổng t.
     + Tăng giá trị j lên 1.
     + Tăng i lên 2.
  4. Kết thúc vòng lặp khi i>2n
  5. Trả về giá trị của t.

4. Hàm main()

1. Nhập giá trị x:
   * Kiểm tra x bằng hàm check(x).
     + Nếu x là số lẻ từ 1 đến 63, thông báo biểu thức A không tồn tại.
     + Ngược lại, tính giá trị A bằng hàm bt1(x) và in ra kết quả.
2. Nhập giá trị n:
   * Tính giá trị B bằng hàm bt2(n) và in ra kết quả.

**II.3 Chương trình nguồn**#include <stdio.h>

#define ll long long

#define db double

int check(db x){

    ll i;

    for(i=1; i<=63; i+=2){

        if(x==i){

            return 1;

            break;

        }

    }

}

db bt1(db x){

    ll i;

    db gt=1;

    for(i=1; i<=63; i+=2){

        gt\*= (1+(-1)/(x-i));

    }

    return gt;

}

ll bt2(ll n){

    ll i=3,j=1 , t=2, temp=2;

    while(i<=2\*n){

        temp= temp\*i\*(i+1)/j;

        t+= temp;

        j++;

        i+=2;

    }

    return t;

}

int main(){

    db x;

    printf("Nhap x: "); scanf("%lf", &x);

    if(check(x)==1){

        printf("Bieu thuc A khong ton tai.\n\n");

    } else {

        printf("Gia tri bieu thuc A= %.2lf\n\n", bt1(x));

    }

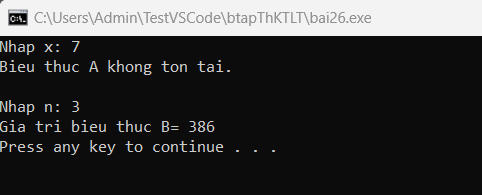
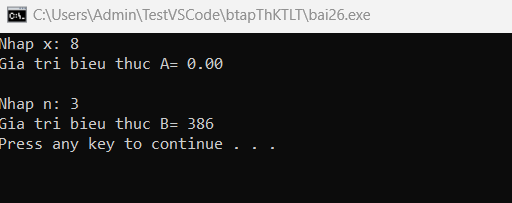
    ll n;

    printf("Nhap n: "); scanf("%lld", &n);

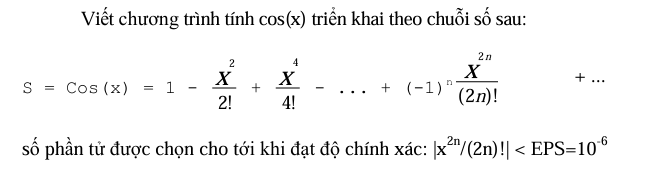
    printf("Gia tri bieu thuc B= %lld\n", bt2(n));

    return 0;

}

**II.4 Kết quả chương trình  
  
**

**III. Câu 48**

**III.1 Tên đề bài**

**III.2 Thuật toán**1. Hàm tinhcos(x)

* Chức năng: Tính giá trị gần đúng của hàm cos(x) sử dụng chuỗi Taylor với sai số eps=10^-6

Thuật toán chi tiết của hàm tinhcos(x):

1. Khởi tạo giá trị:
   * temp = 1 (biến lưu giá trị của từng số hạng trong chuỗi Taylor).
   * t = 1 (tổng giá trị ban đầu của chuỗi, tương ứng với số hạng đầu tiên 1).
   * n = 2 (chỉ số đại diện cho cấp số hạng đầu tiên của chuỗi Taylor, bắt đầu từ x2/2!).
2. Tính chuỗi Taylor:
   * Vòng lặp thực hiện tính từng số hạng của chuỗi với điều kiện dừng: fabs(temp)≥eps, trong đó:
     + fabs(temp) tính giá trị tuyệt đối của số hạng.
     + eps=10^-6, là sai số cho phép.
   * Tại mỗi bước:
     + Cập nhật giá trị của số hạng: temp=temp \* (-x^2)/((n\*(n-1)) (dựa trên mối liên hệ đệ quy giữa các số hạng trong chuỗi Taylor).
     + Cộng số hạng mới vào tổng: t=t+temp
     + Tăng giá trị n lên 2 (chỉ số tiến đến số hạng tiếp theo).
3. Trả về kết quả:
   * Sau khi vòng lặp kết thúc, trả về giá trị gần đúng của cos(x)lưu trong biến t.

2. Hàm main()

1. Nhập giá trị x từ người dùng:
   * Đọc giá trị góc x (đơn vị là độ).
   * Chuyển đổi x từ độ sang radian: x=x×π/180
2. Gọi hàm tinhcos(x) để tính giá trị:
   * Tính giá trị cos(x) sử dụng hàm Taylor đã triển khai.
   * In kết quả tính được.
3. So sánh với giá trị chuẩn:
   * Sử dụng hàm chuẩn cos(x) từ thư viện math.h để tính giá trị chính xác của cos(x)
   * In giá trị từ hàm chuẩn để so sánh.

**III.3 Chương trình nguồn**#include <stdio.h>

#include <math.h>

#define db double

#define ll long long

#define eps 1e-6

db tinhcos(db x) {

    db temp = 1, t = 1;

    ll n = 2;

    while (fabs(temp) >= eps) {

        temp \*= (-1.0 \* x \* x) / (n \* (n - 1));

        t += temp;

        n += 2;

    }

    return t;

}

int main() {

    db x;

    printf("Nhap so do x= ");

    scanf("%lf", &x);

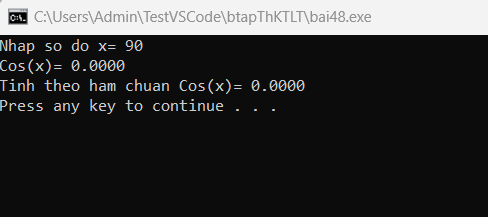
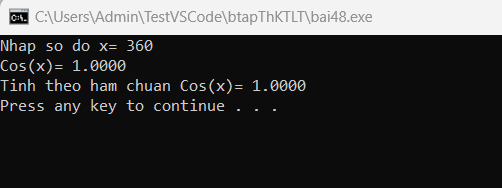
    x = x \* M\_PI / 180.0;

    printf("Cos(x)= %.4lf\n", tinhcos(x));

    printf("Tinh theo ham chuan Cos(x)= %.4lf\n", cos(x));

    return 0;

}

**III.4 Kết quả chương trình  
  
**

**IV.Câu 57**

**IV.1 Tên đề bài**Viết chương trình có sử dụng hàm

a. Viết chương trình nhập mảng một chiều A với n phần tử (n>=10). Xuất mảng A ra màn hình .

b. Xóa các số hoàn hảo trong A trên, xuất lại mảng A sau khi xóa các số hoàn hảo**.**

**IV.2 Thuật toán**1. Hàm nhap(a, n)

* Chức năng: Nhập giá trị cho một mảng số nguyên a có n phần tử từ bàn phím.
* Thuật toán:
  1. Lặp từ 0 đến n−1:
     + Yêu cầu người dùng nhập giá trị cho phần tử thứ i+1.
     + Lưu giá trị nhập vào mảng a.

2. Hàm xuat(a, n)

* Chức năng: In ra toàn bộ giá trị của mảng a có n phần tử.
* Thuật toán:
  1. Lặp từ 0 đến n−1:
     + In giá trị của phần tử a[i].

3. Hàm check(a)

* Chức năng: Kiểm tra xem số nguyên a có phải là số hoàn hảo hay không.
  + Số hoàn hảo là số có tổng các ước (không bao gồm chính nó) bằng chính số đó.
* Thuật toán:
  + Khởi tạo biến sum = 0 để lưu tổng các ước của a.
  + Lặp qua tất cả các số từ 1 đến a/2 (vì không cần kiểm tra lớn hơn a/2):
    - Nếu i là ước của a (tức a%i==0), cộng i vào sum.
  + Sau khi lặp xong:
    - Nếu sum==a, trả về 1 (đúng, a là số hoàn hảo).
    - Nếu không, trả về 0.

4. Hàm xuatshh(a, n)

* Chức năng: Tìm và in ra tất cả các số hoàn hảo trong mảng a.
* Thuật toán:
  1. Lặp qua từng phần tử của mảng a:
     + Kiểm tra từng phần tử a[i] bằng cách gọi hàm check(a[i]).
     + Nếu kết quả là 1 (số hoàn hảo), in giá trị a[i]

5. Hàm main()

1. Nhập giá trị n từ người dùng:
   * Nếu n<10, yêu cầu nhập lại cho đến khi n≥10.
2. Khởi tạo mảng a có kích thước n.
3. Gọi hàm nhap(a, n) để nhập giá trị cho mảng.
4. Gọi hàm xuat(a, n) để in toàn bộ mảng.
5. Gọi hàm xuatshh(a, n) để tìm và in ra tất cả các số hoàn hảo trong mảng.

**IV.3 Chương trình nguồn**#include <stdio.h>

#define ll long long

void nhap(ll a[],ll n){

    ll i;

    printf("Nhap mang A: ");

    for(i=0; i<n; i++){

        printf("A[%lld]= ", i+1);

        scanf("%lld", &a[i]);

    }

}

void xuat(ll a[], ll n){

    ll i;

    printf("Mang A la: ");

    for(i=0; i<n; i++){

        printf("%lld ", a[i]);

    }

    printf("\n");

}

int check(ll a) {

    ll i, sum = 0;

    for ( i = 1; i <= a / 2; i++) {

        if (a % i == 0) {

            sum += i;

        }

    }

    return sum==a;

}

void xuatshh(ll a[], ll n){

    ll i;

    printf("So hoan hao trong mang A la: ");

    for(i=0; i<n; i++){

        if(check(a[i])) printf("%lld ", a[i]);

    }

    printf("\n");

}

int main(){

    ll n;

    printf("Nhap n= "); scanf("%lld", &n);

    while(n<10){

        printf("So n nhap vao khong hop le.\nNhap lai n= ");

        scanf("%lld", &n);

    }

    ll a[n];

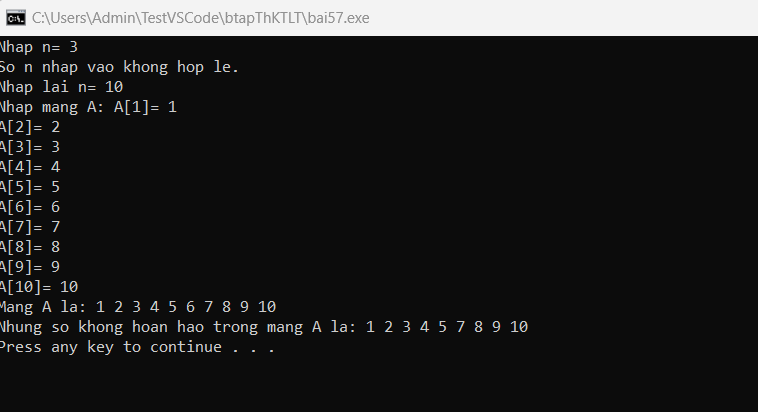
    nhap(a, n);

    xuat(a, n);

    xuatshh(a, n);

    return 0;

}

**IV.4 Kết quả chương trình  
**