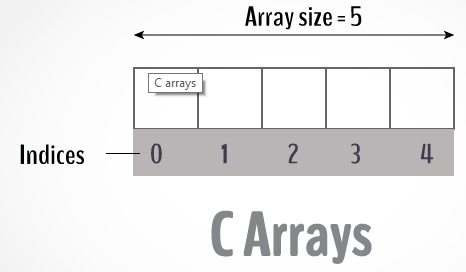
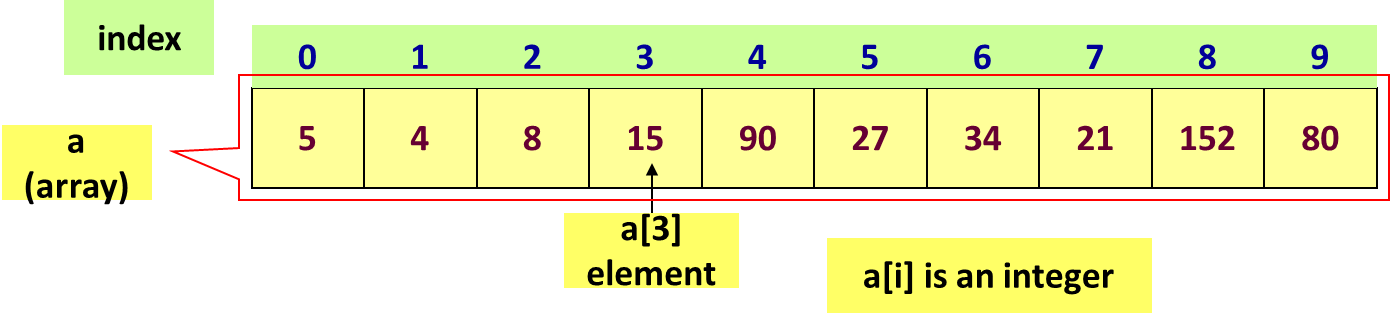
**BÀI THỰC HÀNH SỐ 10: MẢNG MỘT CHIỀU**

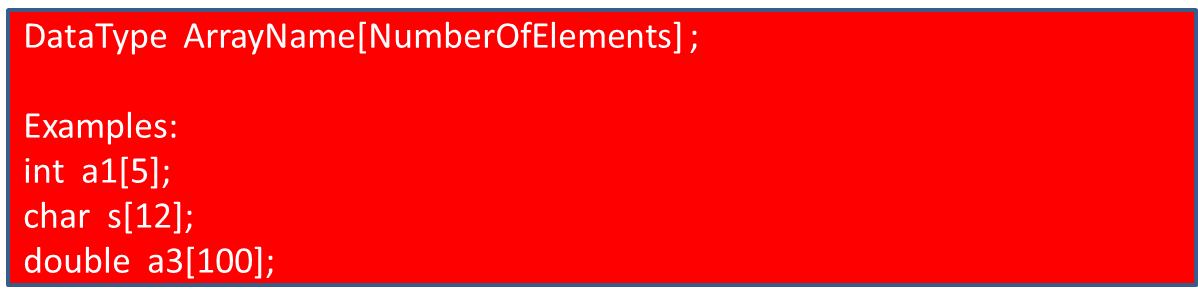
(One-dimensional Arrays)



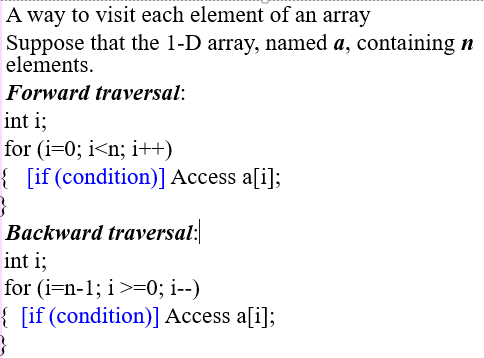
**1-D array:** tập các phần tử cùng kiểu dữ liệu và được lưu trữ liên tiếp trong bộ nhớ.



1. **Khai báo**

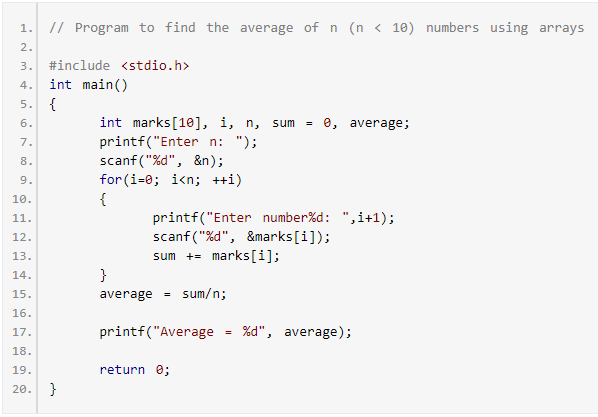


1. **Duyệt mảng**



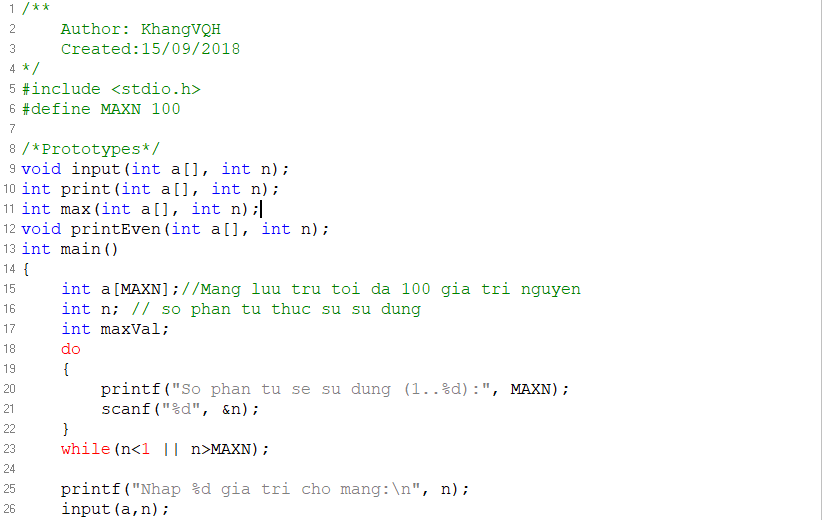
1. **Một số ví dụ**

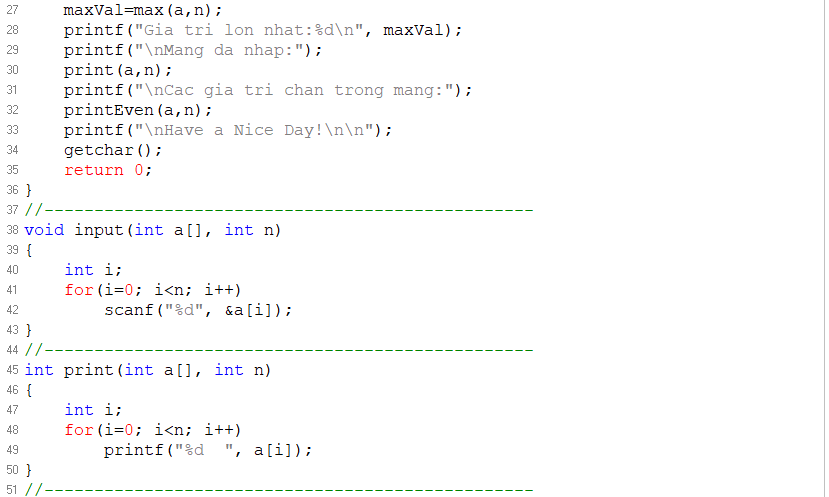
***Example 1:***

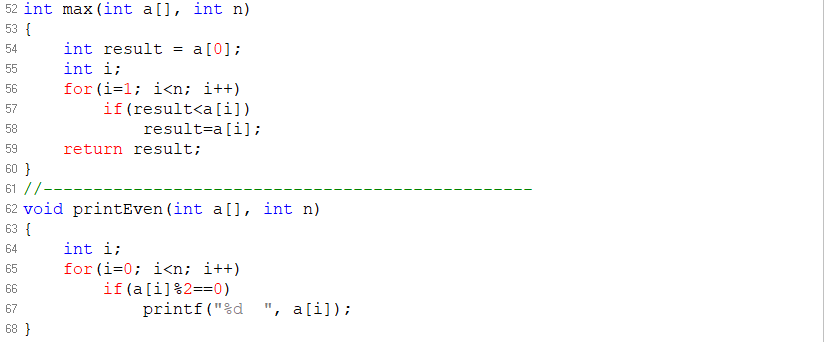


***Example 2:* Chương trình sau thực hiện các yêu cầu:**

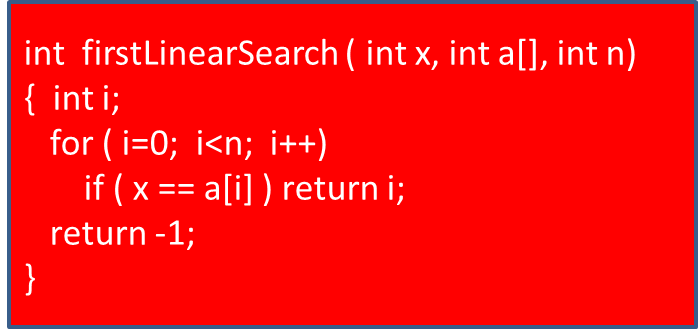
* Nhập các giá trị cho một mảng nguyên có thể chứa tối đa 100 phần tử
* In ra giá trị lớn nhất
* In ra các giá trị của mảng
* In ra các giá trị chẵn.

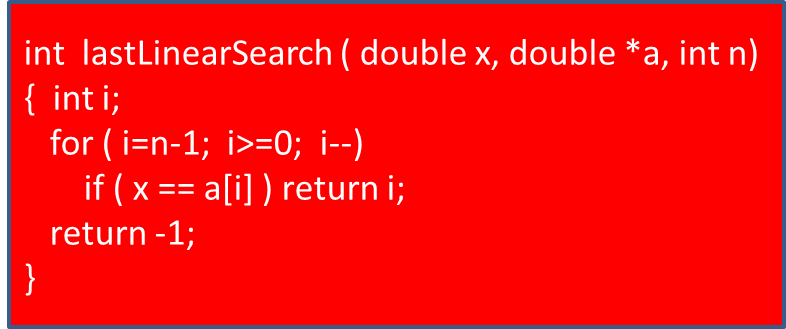


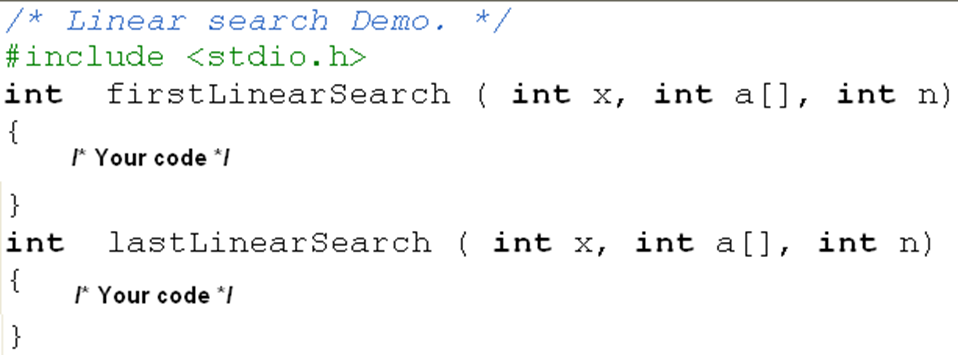
****

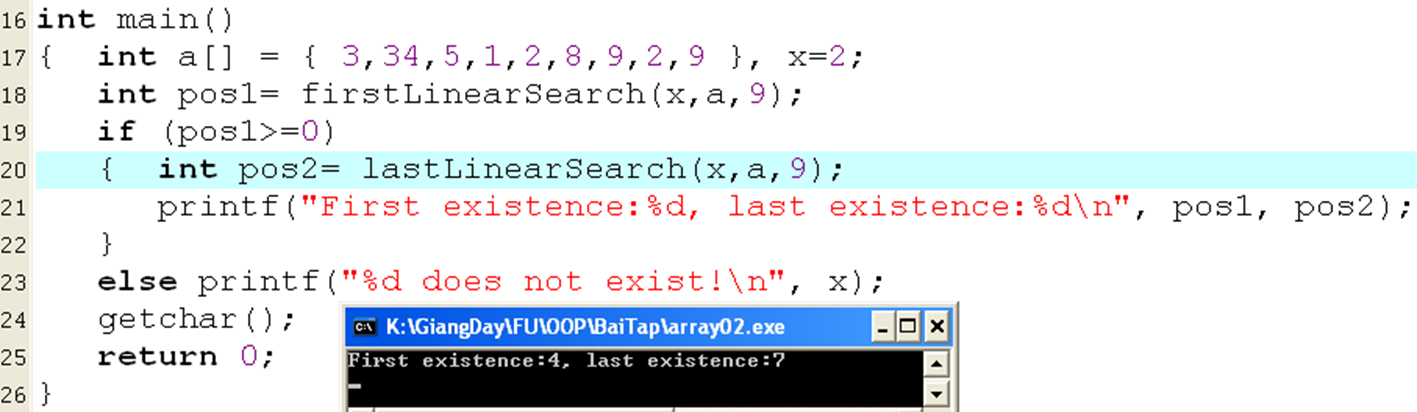
****

***Example 3:* Tìm kiếm trên mảng một chiều**

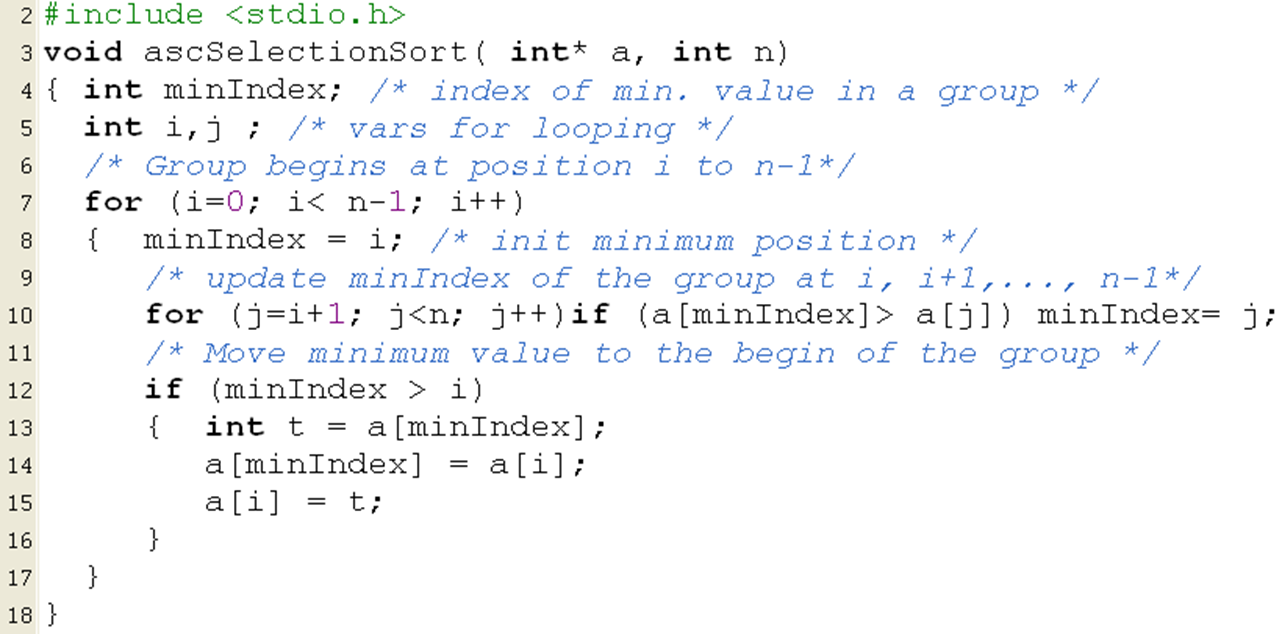


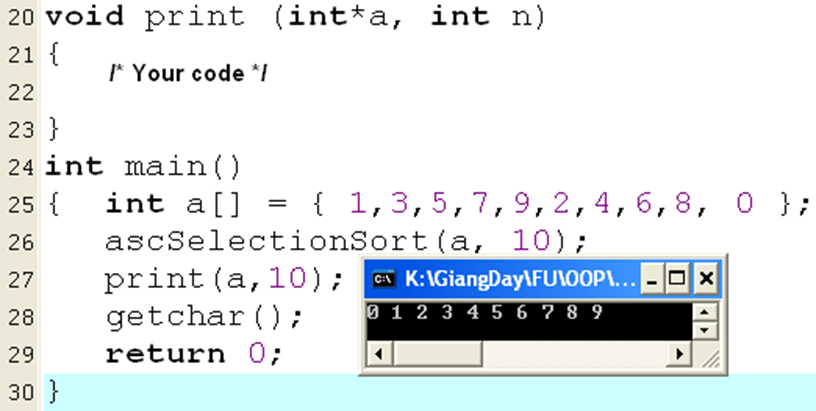




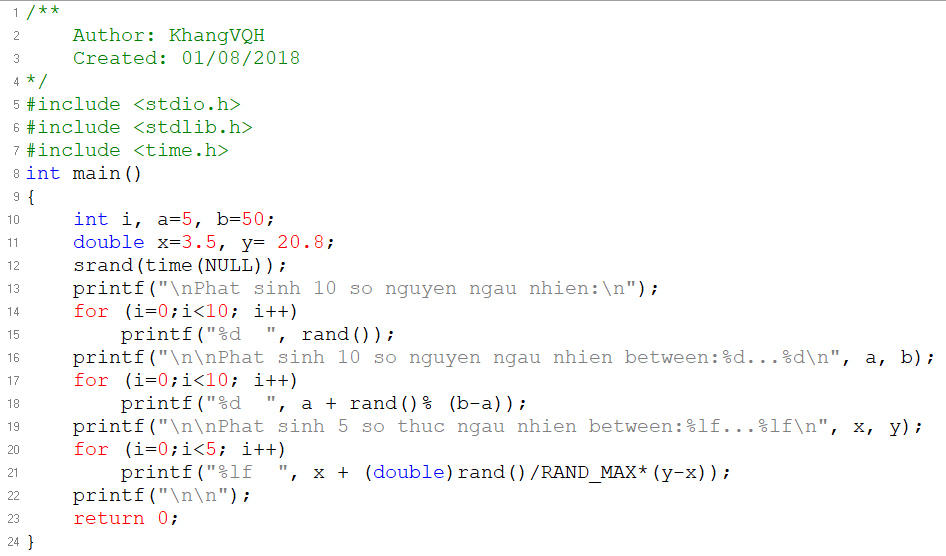


***Example 4:* Sắp xếp mảng một chiều**





***Example 5: Phát sinh ngẫu nhiên Randomize ( stdlib.h )***



1. **BÀI TẬP THỰC HÀNH**

***Bài 1***

Sử dụng mảng một chiều **X[100]** để lưu các số thực. Viết các hàm thực hiện các yêu cầu sau:

1. Nhập n số thực từ bàn phím (0< n ≤100).
2. Hiển thị các giá trị đã nhập ra màn hình.
3. Tính giá trị trung bình của mảng: m = ( x0 + x1 + x2 + ... + xn-1 ) / n.
4. Tính tổng bình phương: ss = x02 + x12 + x22 + ... + xn-12 .
5. Tính phương sai (Variance) d2 = ( ss / n ) - m2.
6. Tính độ lệch chuẩn (Standard deviation): d = sqrt( ( ss / n ) - m2 ).

**Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void InputArr(float X[], int n)

{

int i;

printf ("Nhap vao gia tri cac phan tu: ");

for(i=0;i<n;i++){

do{

printf ("\nX[%d]: ",i);

scanf("%f",&X[i]);

}

while(X[i]<=0||X[i]>100);

}

}

void OutputArr (float X[], int n){

int i;

for(i=0;i<n;i++){

printf ("\nX[%d]: %f",i,X[i]);

}

}

float AVG( float X[], int n){

int i;

float m=0;

for(i=0;i<n;i++){

m+=X[i];

}

return m/n;

}

float Sumpow( float X[], int n){

int i;

float ss=0;

for(i=0;i<n;i++){

ss= pow(X[i],2);

}

return ss;

}

float Var (float X[], int n){

int i;

float dd=0,d=0;

for(i=0; i<n;i++){

dd = sqrt((Sumpow(X,n)/n)-pow(AVG(X,n),2));

}

d = dd\*dd;

return d;

}

float Stand( float X[], int n){

int i;

float d= sqrt((Sumpow(X,n)/n)-pow(AVG(X,n),2));

return d;

}

int main (){

int n;

float X[n];

printf ("Nhap vao so phan tu cua mang: ");

scanf("%d",&n);

InputArr(X,n); //a

OutputArr(X,n); //b

// printf ("\nAVG %f",AVG(X,n));

printf ("\nGia tri do lech chuan : %lf",Stand(X,n));

}

***Bài 2***

Viết chương trình sử dụng mảng 1-D để quản lý một dãy số nguyên (tối đa 100 phần tử), thực hiện các công việc sau:

1. Nhập N giá trị cho mảng.
2. Xuất các phần tử trong mảng.
3. Xuất các số nguyên tố có trong mảng.
4. Xuất các phần tử có giá trị nằm trong khoảng [a, b]. (a, b nhập từ bàn phím)
5. Xuất các phần tử từ vị trí x đến y (x, y nhập từ bàn phím)
6. Tổng các số chẵn.
7. Thêm giá trị X vào vị trí P (X, P nhập từ bàn phím)
8. Xóa phần tử tại ví trí P (P nhập từ bàn phím)
9. Tìm kiếm giá trị X trong mảng (X nhập từ bàn phím)
10. Sắp xếp mảng tăng/giảm

**Code:**

#include <stdio.h>

#include <conio.h>

#include <math.h>

void InputArr(int a[], int n)

{

int i;

printf ("Nhap vao gia tri cac phan tu: \n");

for(i=0;i<n;i++){

scanf("%d",&a[i]);

}

}

void OutputArr (int a[], int n){

int i;

for(i=0;i<n;i++){

printf ("\na[%d]: %d",i,a[i]);

}

}

int CheckSNT(int a){

int i;

if(a<2){

return 0;

}

else{

for(i=2;i<=a/2;i++){

if(a%i==0)

return 0;

}

return 1;

}

}

void SNT(int a[], int n){

printf ("\n Cac so nguyen to trong mang la: \n");

int i;

for (i=0;i<n;i++){

if(CheckSNT(a[i])==1){

printf ("%5d",a[i]);

}

}

}

void Inkhoang( int a[], int n){

int i,x,y;

printf ("\nNhap vao khoang [a-b]: ");

scanf("%d%d",&x,&y);

for (i=0;i<n;i++){

if(a[i]>=x&&a[i]<=y){

printf("\na[%d]: %d",i,a[i]);

}

}

}

void InPt(int a[], int n){

int i,x,y;

printf ("\nNhap vao vi tri x va y: ");

scanf("%d%d",&x,&y);

for(i=0;i<n;i++){

if(i>=x&&i<=y){

printf ("\na[%d]: %d",i,a[i]);

}

}

}

void Sumeven(int a[], int n){

int i,S=0;

for(i=0;i<n;i++){

if(a[i]%2==0){

S+=a[i];

}

}

printf ("\nTong cac so chan trong mang la: %d",S);

}

void AddArr( int a[], int &n){

int x,i,y;

printf ("\nNhap vao vi tri muon them P va gia tri X: ");

scanf("%d%d",&x,&y);

for(i=n;i>x;i--){

a[i]=a[i-1];

}

a[x]= y;

n++;

OutputArr(a,n);

}

void DeleteArr(int a[], int &n){

int i,x;

printf ("\nNhap vao vi tri can xoa");

do{

scanf("%d",&x);

}

while(x<0);

for (i=x;i<n;i++){

a[i]=a[i+1];

}

n--;

OutputArr(a,n);

}

int FindArr(int a[], int n, int b){

int i;

for (i=0;i<n;i++){

if(a[i]==b){

return i;

}

}

return -1;

}

void CheckFind(int a[], int n){

int b;

printf("\nNhap vao gia tri can tim: ");

scanf("%d",&b);

if(FindArr(a,n,b)==-1){

printf("\nKhong tim thay gia tri trong mang");

}

else{

printf ("\nTim thay gia tri tai a[%d]",FindArr(a,n,b));

}

}

int SoftArr\_ascending(int a[], int n){

int i,j;

for( i=0; i< n-1; i++)

{

for(j=i+1; j< n; j++)

if(a[i]>a[j])//neu a[i] > a[j]

{

//dung bien temp dung de doi cho a[i] voi a[j]

int temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

printf ("\nGia tri tang dan: \n");

for (i=0;i<n;i++){

printf ("\na[%d]: %d",i,a[i]);

}

}

int SoftArr\_descending(int a[], int n){

int i,j;

for( i=0; i< n-1; i++)

{

for(j=i+1; j< n; j++)

if(a[i]<a[j])//neu a[i] > a[j]

{

//dung bien temp dung de doi cho a[i] voi a[j]

int temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

printf ("\nGia tri giam dan: \n");

for (i=0;i<n;i++){

printf ("\na[%d]: %d",i,a[i]);

}

}

int main (){

int n,a[n];

printf ("\nNhap vao so phan tu cua mang: ");

scanf("%d",&n);

InputArr(a,n);

OutputArr(a,n);

SNT(a,n);

Inkhoang(a,n);

InPt(a,n);

Sumeven(a,n);

AddArr(a,n);

DeleteArr(a,n);

CheckFind(a,n);

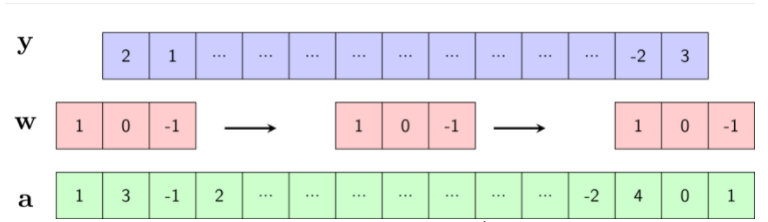
SoftArr\_ascending(a,n);

SoftArr\_descending(a,n);

}

***Bài 3***

**Tính tích chập một chiều (convolution):** cách tính tín hiệu đầu ra **y** được minh họa trong hình sau:



Quá trình tính đầu ra y có thể được thực hiện như sau:

1. Đặt bộ lọc **w** vào vị trí tương ứng với **f** phần tử đầu tiên của **a**.
2. Nhân từng phần tử tương ứng của **w** và **a** rồi cộng các kết quả lại để được phẩn tử tương ứng của **y**.
3. Trượt bộ lọc **w** một bước sang bên phải. Nếu phần tử cuối cùng của bộ lọc không vượt ra ngoài phần tử cuối cùng của tín hiệu, quay lại Bước 2. Ngược lại, dừng các bước tính toán.

Trong ví dụ này:

Viết chương trình nhập vào a, w. Tính y.

***Bài 4***

Viết chương trình thực hiện những yêu cầu sau:

1. Tạo ngẫu nhiên mảng một chiều n phần tử nguyên có giá trị thuộc [-50, 50]
2. Xuất mảng ra màn hình.
3. Xuất các số nguyên tố có trong mảng.
4. Xuất các phần tử có giá trị nằm trong khoảng [a, b]. (a, b nhập từ bàn phím)
5. Xuất các phần tử từ vị trí x đến y (x, y nhập từ bàn phím)
6. Tính trung bình các giá trị của mảng.
7. Đếm số phần tử chia hết cho 4 và có chữ số tận cùng là 6.
8. Thay các phần tử lẻ bằng 2 lần giá trị của nó.
9. Thêm giá trị X vào vị trí P (X, P nhập từ bàn phím)
10. Xóa phần tử tại ví trí P (P nhập từ bàn phím)
11. Tìm kiếm giá trị X trong mảng (X nhập từ bàn phím)

***Bài 5***

Cho hai mảng A, B là hai tập hợp, khởi tạo trước hoặc nhập từ bàn phím. Tạo

