DATA STRUCTURES AND ALGORITHMS IN C++

**Decimal and Binary Conversion Program**

#include<iostream>

using namespace std;

int binaryToDecimal(int number){

    int decimal = 0, pow = 1, rem;

    while(number > 0){

        rem = number % 10;

        number = number / 10;

        decimal += (rem\*pow);

        pow \*= 2;

    }

    return decimal;

}

int decimalToBinary(int number){

    int binary = 0, pow = 1, rem;

    while(number > 0){

        rem = number % 2;

        number = number / 2;

        binary += (rem\*pow);

        pow \*= 10;

    }

    return binary;

}

int main(){

    int binNum,decNum=0;

    cout<<"--CONVERSION PROGRAM--"<<endl;

    cout<<"Enter a decimal number to convert into binary : ";

    cin>>decNum;

    cout<<"Enter a binary number to convert into decimal : ";

    cin>>binNum;

    cout<<"Binary of "<<decNum<<" is : "<<decimalToBinary(decNum)<<endl;

    cout<<"Decimal of "<<binNum<<" is : "<<binaryToDecimal(binNum)<<endl;

    return 0;

}

**Bitwise Operators Program**

#include<iostream>

using namespace std;

int main(){

    cout<<"--BITWISE OPERATORS PRACTICE--"<<endl;

    int num1,num2,numOfShift;

    cout<<"Enter first number to apply bitwise operators : ";

    cin>>num1;

    cout<<"Now Enter second number to apply bitwise operators : ";

    cin>>num2;

    cout<<"Bitwise & : "<<(num1 & num2)<<endl;

    cout<<"Bitwise | : "<<(num1 | num2)<<endl;

    cout<<"Bitwise ^ : "<<(num1 ^ num2)<<endl;

    cout<<"Now Enter the number of shifting : ";

    cin>>numOfShift;

    cout<<numOfShift<<" Left Shift(s) "<<num1<<" is "<<(num1 << numOfShift)<<endl;

    cout<<numOfShift<<" Right Shift(s) "<<num1<<" is "<<(num1 >> numOfShift)<<endl;

    cout<<numOfShift<<" Left Shift(s) "<<num2<<" is "<<(num2 << numOfShift)<<endl;

    cout<<numOfShift<<" Right Shift(s) "<<num2<<" is "<<(num2 >> numOfShift)<<endl;

    return 0;

}

**Checking Number is power of 2?**

#include<iostream>

using namespace std;

int main(){

    int num;

    cout<<"Finding a number that it is a power of 2 or not\n";

    cout<<"Enter a number to check : ";

    cin>>num;

    if(num == 0 || (num % 2) != 0){

        cout<<"Number is not in power of 2.\n";

    }

    else{

        int check = (num & (num-1));

        if( check == 0)

            cout<<"Number is in power of 2.\n";

        else

            cout<<"Number is not in power of 2.\n";

    }

    return 0;

}

**Reversing an input number**

#include<iostream>

using namespace std;

int main(){

    int num,reverse=0,actualNum;

    cout<<"Enter a number to reverse that number : ";

    cin>>num;

    actualNum = num;

    while(num > 0){

        int lastDigit = num % 10;

        reverse = reverse \* 10 + lastDigit;

        num /= 10;

    }

    cout<<"Reverse of "<<actualNum<<" is "<<reverse;

    return 0;

}

**Armstrong Number Checking**

#include<iostream>

#include<math.h>

using namespace std;

int main(){

    int num,sum = 0, originalNum;

    cout<<"Enter a number to check it is ArmStrong number or not : ";

    cin>>num;

    originalNum = num;

    while(num > 0){

        int lastDigit = num % 10;

        sum += pow(lastDigit,3);

        num = num / 10;

    }

    if(sum == originalNum){

        cout<<"Number is Armstrong.\n";

    }

    else{

        cout<<"Number is not Armstrong.\n";

    }

    return 0;

}

**Array Operations Program**

#include<iostream>

using namespace std;

void smallestAndLargestInArray(int numbers[],int size){

    int smallest=numbers[0],largest=numbers[0];

    for(int i=1; i<size; i++){

        smallest = min(numbers[i],smallest);

        largest = max(numbers[i],largest);

    }

    cout<<"Largest Number in Array is: "<<largest;

    cout<<"\nSmallest Number in array is: "<<smallest;

}

int linearSearch(int numbers[],int size,int key){

    for(int i=0; i<size; i++){

        if(numbers[i]==key)

            return i;

    }

    return -1;

}

void reverseArray(int numbers[],int size){

    int start = 0, end = size-1;

    while(start < end){

        swap(numbers[start],numbers[end]);

        start++;

        end--;

    }

}

void swapMinMaxInArray(int numbers[],int size){

    int smallest=numbers[0],largest=numbers[0];

    int smLoc,lgLoc=0;

    for(int i=0; i<size; i++){

        if(smallest>numbers[i]){

            smallest = numbers[i];

            smLoc = i;

        }

        if(largest<numbers[i]){

            largest = numbers[i];

            lgLoc = i;

        }

    }

    int temp = numbers[smLoc];

    numbers[smLoc] = numbers[lgLoc];

    numbers[lgLoc] = temp;

}

void printUniqueInArray(int numbers[],int size){

    int count;

    cout<<"\nUnique Elements in array : ";

    for(int i=0; i<size; i++){

        count = 0;

        for(int j=0; j<size; j++){

            if(numbers[i]==numbers[j])

                count++;

        }

        if(count<2)

            cout<<numbers[i]<<" ";

    }

    cout<<endl;

}

void intersectArrays(int a[],int b[],int sizeA,int sizeB){

    cout<<"\nIntersection of Arrays are : ";

    for(int i=0; i<sizeA; i++){

        for(int j=0; j<sizeB; j++){

            if(a[i]==b[j]){

                cout<<a[i]<<" ";

            }

        }

    }

    cout<<endl;

}

int main(){

    int numbers[] = {2,4,6,8,10,12,14,16,18,20};

    int numbers2[] = {2,4,6,9,10};

    int redundantNums[] = {1,4,3,4,7,8,8,7,3,9,5,6,10};

    smallestAndLargestInArray(numbers,10);

    int location = linearSearch(numbers,10,20);

    if(location != -1)

        cout<<"\nKey Number found at location: "<<location<<endl;

    else

        cout<<"\nElement not found.\n";

    cout<<"Array Before Reversing...\n";

    for(int i=0; i<10; i++)

        cout<<" "<<numbers[i]<<"  ";

    cout<<"\nArray After Reversing...\n";

    reverseArray(numbers,10);

    for(int i=0; i<10; i++)

        cout<<" "<<numbers[i]<<"  ";

    swapMinMaxInArray(numbers,10);

    cout<<"\nArray After Swapping Minimum and Maximum number in array : ";

    for(int i=0; i<10; i++)

        cout<<" "<<numbers[i]<<"  ";

    printUniqueInArray(redundantNums,13);

    intersectArrays(numbers,numbers2,10,5);

    return 0;

}

**Vector Operations Program**

#include<iostream>

#include<vector>

using namespace std;

void reverseVector(vector <int> &vec,int size){

    int start = 0;

    int end = size-1;

    while(start<end){

        swap(vec.at(start),vec.at(end));

        start++;

        end--;

    }

}

int main(){

//  vector <int> vec(3,4);

    vector <int> vec;

    for(int i=1; i<=50; i++){

        vec.push\_back(i);

    }

    cout<<"Vector Values after pushing 50 numbers: ";

    for(int i=0; i<50; i++){

        cout<<vec.at(i)<<"  ";

    }

    for(int i=0; i<25; i++){

        vec.pop\_back();

    }

    cout<<"\nVector after poping 25 values: ";

    for(int i=0; i<25; i++){

        cout<<vec.at(i)<<"  ";

    }

    cout<<"\nFront Value of the vector is: "<<vec.front();

    cout<<"\nBack Value of the vector is: "<<vec.back();

    cout<<"\nSize of the Vector is: "<<vec.size();

    cout<<"\nCapacity of the vector is: "<<vec.capacity();

    reverseVector(vec,vec.size());

    cout<<"\nVector Values after Reversing: ";

    for(int i=0; i<vec.size(); i++){

        cout<<vec.at(i)<<"  ";

    }

    return 0;

}

**Single Number Problem**

#include<iostream>

using namespace std;

int findSingleNumber(int nums[],int size){

    int singleNumber=nums[0];

    for(int i=1; i<size; i++){

        singleNumber ^= nums[i];

    }

    return singleNumber;

}

int main(){

    int numbers[9]={1,2,2,1,3,4,4,5,3};

    cout<<"Single Number in array is: "<<findSingleNumber(numbers,9);

    return 0;

}

**Maximum Sub Array Sum Problem**

#include<iostream>

using namespace std;

void subArrays(int numbers[],int size){

    cout<<"Sub arrays are: \n";

    for(int st=0; st<size; st++){

        for(int end=st; end<size; end++){

            for(int i=st; i<=end; i++){

                cout<<numbers[i];

            }

            cout<<" ";

        }

        cout<<endl;

    }

}

int maxSubArray(int numbers[],int size){

    int maxSum = INT\_MIN;

    for(int st=0; st<size; st++){

        int crntSum = 0;

        for(int end=st; end<size; end++){

            crntSum += numbers[end];

            maxSum = max(crntSum,maxSum);

        }

    }

    return maxSum;

}

int kadanesMaxSubArray(int numbers[],int size){

    int maxSum = INT\_MIN, crntSum = 0;

    for(int i=0; i<size; i++){

        crntSum += numbers[i];

        maxSum = max(crntSum,maxSum);

        if(crntSum<0)

            crntSum = 0;

    }

    return maxSum;

}

int main(){

    int numbers[]={1,2,-3,-4,5};

    subArrays(numbers,5);

    cout<<"\nMaximum sub array sum is (Without using any Algorithm): "<<maxSubArray(numbers,5);

    cout<<"\nMaximum sub array sum is (By Using Kadanes Algorithm): "<<kadanesMaxSubArray(numbers,5);

    return 0;

}

**Pair Sum Problem**

#include<iostream>

#include<vector>

using namespace std;

vector<int> pairSum(vector<int> &vec,int target){

    vector <int> targetIndices;

    for(int i=0; i<vec.size(); i++){

        for(int j=i+1; j<vec.size(); j++){

            if(vec.at(i)+vec.at(j)==target){

                targetIndices.push\_back(i);

                targetIndices.push\_back(j);

                return targetIndices;

            }

        }

    }

}

vector<int> pairSumUsing2ptrs(vector<int> &vec,int target){

    vector <int> targetIndices;

    int i=0,j=vec.size()-1,pairSum=0;

    while(i<j){

        pairSum = vec.at(i) + vec.at(j);

        if(pairSum>target)  j--;

        else if(pairSum<target) i++;

        else{

            targetIndices.push\_back(i);

            targetIndices.push\_back(j);

            return targetIndices;

        }

    }

}

int main(){

    vector <int> vec,targetIndices;

    cout<<"Pair Sum Program\n";

    cout<<"Vector: ";

    for(int i=1; i<=10; i++)

        vec.push\_back(i\*2);

    for(int i=0; i<10; i++)

        cout<<"["<<i<<"]: "<<vec.at(i)<<"  ";

    targetIndices = pairSum(vec,22);

    cout<<"\nTarget (22) is the sum of the numbers at indices(Using Brutte Force Approach) : ";

    for(int i=0; i<targetIndices.size(); i++)

        cout<<targetIndices.at(i)<<" ";

    cout<<"\nTarget (12) is the sum of the numbers at indices(Using 2 Pointers Approach) : ";

    targetIndices = pairSumUsing2ptrs(vec,12);

    for(int i=0; i<targetIndices.size(); i++)

        cout<<targetIndices.at(i)<<" ";

    return 0;

}

**Majority Element Problem**

#include<iostream>

#include<vector>

using namespace std;

int majorityElement(vector<int> &vec){

    for(int i=0; i<vec.size(); i++){

        int freq=0;

        for(int j=0; j<vec.size(); j++){

            if(vec.at(i)==vec.at(j)) freq++;

        }

        if(freq>vec.size()/2){

            return vec.at(i);

        }

    }

}

//int optimizedMajorityElement(vector<int> &vec){

//  int n = vec.size(),freq = 1, me = vec.at(0);

//  sort(vec.begin(),vec.end());

//  for(int i=1; i<n; i++){

//      if(vec.at(i)==vec.at(i-1)) freq++;

//      else{

//          freq = 1;

//          me = vec.at(i);

//      }

//      if(freq>n/2) return me;

//  }

//  return me;

//}

int mooresMajorityElement(vector<int> &vec){

    int freq=0,me=0;

    for(int i=0; i<vec.size(); i++){

        if(freq == 0) me = vec.at(i);

        if(me==vec.at(i)) freq++;

        else freq--;

    }

    return me;

}

int main(){

    vector<int> numbers(3,4);

    numbers.push\_back(2);

    numbers.push\_back(3);

    numbers.push\_back(4);

    cout<<"Majority Element in the array/vector is: "<<majorityElement(numbers);

//  cout<<"\nMajority Element(Optimized Approach) in the array/vector is: "<<optimizedMajorityElement(numbers);

    cout<<"\nMajority Element(Using Moore's Algorithm) in the array/vector is: "<<mooresMajorityElement(numbers);

    return 0;

}