Programs in c

1. Armstrong number

#include <stdio.h>

int main() {

int num, originalNum, remainder, result = 0;

printf("Enter a three-digit integer: ");

scanf("%d", &num);

originalNum = num;

while (originalNum != 0) {

// remainder contains the last digit

remainder = originalNum % 10;

result += remainder \* remainder \* remainder;

// removing last digit from the orignal number

originalNum /= 10;

}

if (result == num)

printf("%d is an Armstrong number.", num);

else

printf("%d is not an Armstrong number.", num);

return 0;

}

Output

Enter a three-digit integer: 371

371 is an Armstrong number.

2.palindrome

#include <stdio.h>

int main() {

int n, reversed = 0, remainder, original;

printf("Enter an integer: ");

scanf("%d", &n);

original = n;

// reversed integer is stored in reversed variable

while (n != 0) {

remainder = n % 10;

reversed = reversed \* 10 + remainder;

n /= 10;

}

// palindrome if orignal and reversed are equal

if (original == reversed)

printf("%d is a palindrome.", original);

else

printf("%d is not a palindrome.", original);

return 0;

}

3.stack

plementation is using arrays, but it can also be implemented using lists.

Python

Java

C

C++

// Stack implementation in C

#include <stdio.h>

#include <stdlib.h>

#define MAX 10

int count = 0;

// Creating a stack

struct stack {

int items[MAX];

int top;

};

typedef struct stack st;

void createEmptyStack(st \*s) {

s->top = -1;

}

// Check if the stack is full

int isfull(st \*s) {

if (s->top == MAX - 1)

return 1;

else

return 0;

}

// Check if the stack is empty

int isempty(st \*s) {

if (s->top == -1)

return 1;

else

return 0;

}

// Add elements into stack

void push(st \*s, int newitem) {

if (isfull(s)) {

printf("STACK FULL");

} else {

s->top++;

s->items[s->top] = newitem;

}

count++;

}

// Remove element from stack

void pop(st \*s) {

if (isempty(s)) {

printf("\n STACK EMPTY \n");

} else {

printf("Item popped= %d", s->items[s->top]);

s->top--;

}

count--;

printf("\n");

}

// Print elements of stack

void printStack(st \*s) {

printf("Stack: ");

for (int i = 0; i < count; i++) {

printf("%d ", s->items[i]);

}

printf("\n");

}

// Driver code

int main() {

int ch;

st \*s = (st \*)malloc(sizeof(st));

createEmptyStack(s);

push(s, 1);

push(s, 2);

push(s, 3);

push(s, 4);

printStack(s);

pop(s);

4.queue

// Queue implementation in C

#include <stdio.h>

#define SIZE 5

void enQueue(int);

void deQueue();

void display();

int items[SIZE], front = -1, rear = -1;

int main() {

//deQueue is not possible on empty queue

deQueue();

//enQueue 5 elements

enQueue(1);

enQueue(2);

enQueue(3);

enQueue(4);

enQueue(5);

// 6th element can't be added to because the queue is full

enQueue(6);

display();

//deQueue removes element entered first i.e. 1

deQueue();

//Now we have just 4 elements

display();

return 0;

}

void enQueue(int value) {

if (rear == SIZE - 1)

printf("\nQueue is Full!!");

else {

if (front == -1)

front = 0;

rear++;

items[rear] = value;

printf("\nInserted -> %d", value);

}

}

void deQueue() {

if (front == -1)

printf("\nQueue is Empty!!");

else {

printf("\nDeleted : %d", items[front]);

front++;

if (front > rear)

front = rear = -1;

}

}

// Function to print the queue

void display() {

if (rear == -1)

printf("\nQueue is Empty!!!");

else {

int i;

printf("\nQueue elements are:\n");

for (i = front; i <= rear; i++)

printf("%d ", items[i]);

}

printf("\n");

}

5.duck number

# include <stdio.h>

# include <stdlib.h>

int main()

{

int dno,dkno,r,flg;

flg=0;

printf("\n\n Check whether a number is a Duck Number or not: \n");

printf(" ----------------------------------------------------\n");

printf(" Input a number: ");

scanf("%d",&dkno);

dno=dkno;

while(dkno>0)

{

if(dkno % 10 == 0)

{

flg=1;

break;

}

dkno/=10;

}

if(dno>0 && flg==1)

{

printf(" The given number is a Duck Number.\n");

}

else

{

printf(" The given number is not a Duck Number.\n");

}

}

6.

#include <stdio.h>

#include <conio.h>

int main ()

{

// declare integer variables

int n, temp, rev = 0, digit, sum\_of\_digits = 0;

printf (" Enter a Number: \n");

scanf (" %d", &n); // get the number

temp = n; // assign the number to temp variable

// use while loop to calculate the sum of digits

while ( temp > 0)

{

// extract digit one by one and store into the sum\_of\_digits

sum\_of\_digits = sum\_of\_digits + temp % 10; /\* use modulus symbol to get the remainder of each iteration by temp % 10 \*/

temp = temp / 10;

}

temp = sum\_of\_digits; // assign the sum\_of\_digits to temp variable

printf (" \n The sum of the digits = %d", temp);

// get the reverse sum of given digits

while ( temp > 0)

{

rev = rev \* 10 + temp % 10;

temp = temp / 10;

}

printf (" \n The reverse of the digits = %d", rev);

printf (" \n The product of %d \* %d = %d", sum\_of\_digits, rev, rev \* sum\_of\_digits);

// use if else statement to check the magic number

if ( rev \* sum\_of\_digits == n)

{

printf (" \n %d is a Magic Number. ", n);

}

else

{

printf (" \n %d is not a Magic Number. ", n);

}

return 0;

}