* XI CODES

🡪 #include <iostream>

using namespace std;

int change(int num)

{

if(num % 2 == 0)

{

return num \* 2;

}

else

{

return num \* 3;

}

}

int main()

{

int a[30];

int n;

cin >> n;

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

{

cin >> a[i];

a[i] = change(a[i]);

}

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

{

cout << a[i] << " ";

}

return 0;

}

🡪 #include <iostream>

using namespace std;

int power(int, int);

int main()

{

int rem;

int quo;

int dec = 0;

int bin;

int i = 0;

cin >> bin;

quo = bin;

while(quo != 0)

{

rem = quo % 10;

dec += rem \* power(2, i);

i++;

quo = quo / 10;

}

cout << dec << endl;

return 0;

}

int power(int base, int exponent)

{

int pro = 1;

for(int i = 1; i <= exponent; i++)

{

pro \*= base;

}

return pro;

}

* #include <iostream>

#include <string>

using namespace std;

int main()

{

string s;

getline(cin, s);

cout << "length : " << s.length() << endl;

if(s.find('ani'))

{

cout << "yes";

}

else

{

cout << "no";

}

return 0;

}

🡪#include <iostream>

#include <string.h>

using namespace std;

//class definition

class Test1

{

int value;

char string[20];

public:

void get()

{

cin >> value;

cin.ignore();

cin.getline(string, 20);

}

void show()

{

cout << value << endl;

cout << string << endl;

}

};

//structure definition

struct Test2

{

int value;

char string[20];

};

//union definition

union Test3

{

int value;

char string[20];

};

int main()

{

//public access

Test1 a;

Test2 b;

Test3 c;

// a.value = 45;

// a.string = "kanishk";

a.get();

a.show();

b.value = 45;

strcpy(b.string, "kanishk");

cout << sizeof(b) << endl;

c.value = 45;

strcpy(c.string, "kanishk");

cout << sizeof(c) << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int count = 0;

int countPresest = 0;

int currency[] = {2000, 500, 200, 100, 50, 20, 10, 5, 2, 1};

int amount;

cout << "Enter amount : "; cin >> amount;

for(int i = 0; i < 10; i++)

{

countPresest = amount / currency[i];

count += amount / currency[i];

if(countPresest != 0)

cout << currency[i] << " notes : " << countPresest << endl;

amount %= currency[i];

}

cout << "Currency notes required = " << count << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

int power(int, int);

int main()

{

int rem;

int quo;

int dec;

int bin = 0;

int i = 0;

cin >> dec;

quo = dec;

while(quo != 0)

{

rem = quo % 2;

bin = rem \* power(10, i) + bin;

i++;

quo = quo / 2;

}

cout << bin << endl;

return 0;

}

int power(int base, int exponent)

{

int pro = 1;

for(int i = 1; i <= exponent; i++)

{

pro \*= base;

}

return pro;

}

🡪#include <iostream>

using namespace std;

float sellingPrice(float costPrice, float discount)

{

return costPrice \* (100 - discount) / 100;

}

int main()

{

float costPrice, discount;

cout << "Cost Price : "; cin >> costPrice;

if(costPrice > 10000)

{

discount = 15.00;

}

else if(costPrice <= 10000 && costPrice > 5000)

{

discount = 10.00;

}

else if(costPrice <= 5000 && costPrice > 1000)

{

discount = 5.00;

}

else

{

discount = 3.00;

}

cout << "Selling Price : " << sellingPrice(costPrice, discount) << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int n;

cin >> n;

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

{

if(n % i == 00)

{

cout << i;

if(i < n)

{

cout <<", ";

}

}

}

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int n;

int a = 0, b = 1;

int c;

cin >> n;

cout << a << " " << b << " ";

for(int i = 0; i < n - 2; i++)

{

c = a + b;

a = b;

b = c;

cout << c << " ";

}

return 0;

}

🡪#include <iostream>

#include <stdio.h>

#include <ctype.h>

using namespace std;

int main()

{

char input[20];

char output[20];

gets(input);

int i;

int k = 0;

for(i = 0; input[i] != NULL; i++)

{

if(isalpha(input[i])) //if(input[i] >= 'a' && input[i] <= 'z' || input[i] >= 'A' && input[i] <= 'Z')

{

output[k] = input[i];

cout << "output string position" << k << " : " << output[k];

k++;

}

cout << endl;

}

output[k] = NULL;

cout << output;

}

🡪#include <iostream>

#define SIZE 40

using namespace std;

int findArray(int a[], int n, int num)

{

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

{

if(a[i] == num)

{

return 1;

}

}

return 0;

}

int main()

{

int a[SIZE];

int n;

int num;

cout << "Enter the size ( < 40) : "; cin >> n;

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

{

cin >> a[i];

}

cout << "Enter the number to be checked : "; cin >> num;

cout << "check = " << findArray(a, n, num) << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

//function prototypes

void function1(void);

void function2(int);

long function3();

long function4(int);

//driver function

int main(void)

{

/\* code \*/

function1();

function2(5);

cout << function3() << endl;

cout << function4(7) << endl;

if(isPalindrome(5465)

{i

std::cout << "yes" << '\n';

}

else

{

std::cout << "no" << '\n';

}

return 0;

}

//function definitions

void function1(void)

{

//printing functions

cout << "this is a normal void function which doesn't take any arguments." << endl;

}

void function2(int n)

{

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

{

cout << i + 1 << endl;

}

}

long function3()

{

long a, b, sum;

cout << "enter a : "; cin >> a;

cout << "enter b : "; cin >> b;

sum = a + b;

return sum;

}

long function4(int n)

{

long fact = 1;

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

{

/\* code \*/

fact \*= i;

}

return fact;

}

🡪#include <iostream>

#include <string.h>

using namespace std;

struct Game

{

long g\_code;

char g\_name[30];

int fee;

int duration;

};

void input(Game &g)

{

cout << "Enter game code : "; cin >> g.g\_code;

cin.ignore();

cout << "Enter game name : "; cin.getline(g.g\_name, 20);

//logic for fee and duration

//g.g\_name == "table tennis"

if(strcmp(g.g\_name, "table tennis") == 0)

{

g.fee = 24000;

g.duration = 3;

}

else if(strcmp(g.g\_name, "swimming") == 0)

{

g.fee = 30000;

g.duration = 1;

}

else if(strcmp(g.g\_name, "football") == 0)

{

g.fee = 25000;

g.duration = 2;

}

else

{

g.fee = 0;

g.duration = 0;

}

}

void output(Game g)

{

cout << "Game code : " << g.g\_code << endl;

cout << "Game name : " << g.g\_name << endl;

cout << "Game fee : " << g.fee << endl;

cout << "Game duration : " << g.duration << endl;

}

int main()

{

Game g[4];

for(int i = 0; i < 4; i++)

{

cout << "Enter details for game "<< i + 1 <<" : " << endl;

input(g[i]);

}

for(int i = 0; i < 4; i++)

{

output(g[i]);

}

return 0;

}

🡪//this project is made by kanishk

//this project helps in checking for a vowel

#include <iostream>

using namespace std;

//function prototype

int isVowel(char);

//main driver function

int main()

{

char ch;

cout << "Enter a character : "; cin >> ch;

if(isVowel(ch))

{

cout << "It's a vowel." << endl;

}

else

{

cout << "No it's not a vowel." << endl;

}

return 0;

}

//function to check for vowel

int isVowel(char ch)

{

//

switch(ch)

{

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

case 'A':

case 'E':

case 'I':

case 'O':

case 'U': return 1;

default : return 0; //takes all other characters

}

}

🡪#include <iostream>

#include <string>

using namespace std;

int main()

{

string s;

char x;

int count;

getline(cin, s);

for(int i = 0; s[i] != NULL; i++)

{

count = 0;

if(s[i] != '\*')

{

x = s[i];

for(int j = 0; s[j] != NULL; j++)

{

if(x == s[j])

{

count++;

s[j] = '\*';

}

}

}

if(count != 0)

cout << x << " : " << count << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int a[10][10], b[10][10], sum[10][10], diff[10][10];

int row, col;

cout <<"Enter number of rows and columns : ";

cin >> row >> col;

cout << "Enter elements of a : " << endl;

for(int i = 0; i < row; i++)

{

for(int j = 0; j < col; j++)

{

cin >> a[i][j];

}

}

cout << "Enter elements of b : " << endl;

for(int i = 0; i < row; i++)

{

for(int j = 0; j < col; j++)

{

cin >> b[i][j];

}

}

for(int i = 0; i < row; i++)

{

for(int j = 0; j < col; j++)

{

sum[i][j] = a[i][j] + b[i][j];

diff[i][j] = a[i][j] - b[i][j];

}

}

cout << "Sum = " << endl;

for(int i = 0; i < row; i++)

{

for(int j = 0; j < col; j++)

{

cout << sum[i][j] << "\t";

}

cout << endl;

}

cout << "Difference = " << endl;

for(int i = 0; i < row; i++)

{

for(int j = 0; j < col; j++)

{

cout << diff[i][j] << "\t";

}

cout << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

void swap(int &a, int &b)

{

int temp;

temp = a;

a = b;

b = temp;

}

int main()

{

int m[5][5];

int row, column;

cout << "Enter row and column (<5 only): "; cin >> row >> column;

for(int i = 0; i < row; i++)

{

for(int j = 0; j < column; j++)

{

cin >> m[i][j];

}

}

for(int j = 0; j < column; j++)

{

swap(m[0][j], m[row - 1][j]);

}

for(int i = 0; i < row; i++)

{

for(int j = 0; j < column; j++)

{

cout << m[i][j] << "\t";

}

cout << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int a[5][5];

int mainSum = 0;

int secSum = 0;

//input of matrix

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

cin >> a[i][j];

}

}

//logic

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

if(i == j)

{

mainSum += a[i][j];

}

if(i + j == 3 - 1)

{

secSum += a[i][j];

}

}

}

cout << "main diagonal sum = " << mainSum << endl;

cout << "secondary diagonal sum = " << secSum << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int a[5][5];

int sum1 = 0;

int sum2 = 0;

int sum3 = 0;

int sum4 = 0;

//input of matrix

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

cin >> a[i][j];

}

}

//logic

for(int i = 0; i < 3; i++)

{

for(int j = 0; j < 3; j++)

{

if(i <= j)

{

sum1 += a[i][j];

}

if(i >= j)

{

sum2 += a[i][j];

}

if(i + j < 3)

{

sum3 += a[i][j];

}

if(i + j >= 2)

{

sum4 += a[i][j];

}

}

}

cout << "sum 1 = " << sum1 << endl;

cout << "sum 2 = " << sum2 << endl;

cout << "sum 3 = " << sum3 << endl;

cout << "sum 4 = " << sum4 << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int n;

cin >> n;

int count = 0;

while(n != 0)

{

count++;

n /= 10;

}

cout << "Number of digits were = " << count << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int s[10];

int x;

int count;

for(int i = 0; i < 10; i++)

{

cin >> s[i];

}

for(int i = 0; i < 10; i++)

{

count = 0;

if(s[i] != -404)

{

x = s[i];

for(int j = 0; j < 10; j++)

{

if(x == s[j])

{

count++;

s[j] = -404;

}

}

}

if(count != 0)

cout << x << " : " << count << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

int g = 20; //let it be k = 20 (changable call by refrence by default)

void function(int &x, int y) //x is refrence and y is value

{

//y will not change until it's static or global

x = x - y; //1. x = -13| 2. x = 33

y = 10 \* x; //1. y = -130| 2. y = 330

cout << x << " " << y << endl;

//-13 -130

//20 330

}

int main()

{

int g = 7; //g = 7, local g

function(g, ::g); //both g's will change g = -13, ::g = -130

cout << g << " " << ::g << endl;

//-13 20

function(::g, g); //::g will change but g will not

cout << g << " " << ::g << endl;

//-13 330

}

//-13 -130

//-13 -130

//117 1170

//-13 117

🡪#include <iostream>

using namespace std;

int a = 40;

void demo(int &x, int y, int z)

{

a += x;

y \*= a;

z = a + y;

cout << x <<'\t'<< y <<'\t' << z << endl;

}

int main()

{

int a = 25, b = 15;

demo(::a, a, b);

cout << ::a <<'\t'<< a <<'\t' << b << endl;

return 0;

}

🡪

int main()

{

int a[10];

for(int i = 0; i < 10; i++)

{

cin >> a[i];

}

cout << "Number of palindromes are : " << palindromeCount(a, 10) << endl;

return 0;

}

int palindromeCount(int a[], int size)

{

int count = 0;

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

{

if(isPalindrome(a[i]))

{

count++;

}

}

return count;

}

int isPalindrome(int num)

{

int r, temp, check = 0;

temp = num;

while(temp != 0)

{

r = temp % 10;

check = check \* 10 + r;

temp /= 10;

}

if(check == num)

{

return 1;

}

else

{

return 0;

}

}

🡪#include <iostream>

#include <string.h>

using namespace std;

void swap(char \*x, char \*y)

{

char temp;

temp = \*x;

\*x = \*y;

\*y = temp;

}

void permute(char \*a, int l, int r)

{

int i;

if (l == r)

cout << a;

else

{

for (i = l; i <= r; i++)

{

swap((a+l), (a+i));

permute(a, l+1, r);

swap((a+l), (a+i)); //backtrack

}

}

cout << endl;

}

int main()

{

char str[] = "ABC";

int n = strlen(str);

permute(str, 0, n-1);

return 0;

}

🡪#include <iostream>

using namespace std;

void swap(int \*, int \*);

int main()

{

int a, b;

a = 45;

b = 89;

cout << "a = " << a << endl;

cout << "b = " << b << endl;

swap(a, b);

cout << "a = " << a << endl;

cout << "b = " << b << endl;

return 0;

}

void swap(int \*a, int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

}

🡪#include <iostream>

using namespace std;

int area(int a)

{

cout << "function 1 called" << endl;

return a \* a;

}

double area(double r)

{

cout << "function 2 called" << endl;

return 3.14 \* r \* r;

}

int main()

{

cout << area(5) << endl;

cout << area(5.6) << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

long factorial(int);

int power(int, int);

int main()

{

int x, n;

float sum = 0;

cin >> x >> n;

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

{

sum += (float)power((-1), i + 1) \* power(x, i) / factorial(i);

}

cout << sum;

return 0;

}

long factorial(int n)

{

if(n == 1)

{

return 1;

}

else

{

return n \* factorial(n - 1);

}

}

int power(int base, int exponent)

{

int pro = 1;

for(int i = 1; i <= exponent; i++)

{

pro \*= base;

}

return pro;

}

🡪#include <iostream>

using namespace std;

int primeCount(int\*, int);

int isPrime(int);

int main()

{

int a[7];

for(int i = 0; i < 7; i++)

{

cin >> a[i];

}

cout << "Total number of prime is : " << primeCount(a, 7) << endl;

return 0;

}

int primeCount(int a[], int n)

{

int count = 0;

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

{

if(isPrime(a[i]) == 1)

{

cout << a[i] << ", ";

count++;

}

}

return count;

}

int isPrime(int num)

{

if(num == 1)

{

return 0;

}

for(int i = 2; i <= num / 2; i++)

{

if(num % i == 0)

{

return 0; //false condition

}

}

return 1; //true condition

}

🡪#include <iostream>

using namespace std;

void strcon(char s[])

{

for(int i = 0, l = 0; s[i] != '\0'; i++, l++)

{

cout << "---------------" << endl;

cout << "i = " << i << " and l = " << l << endl;

for(int j = 0; j < l; j++)

{

cout << "before : " << s[j] << " ----- ";

if(isupper(s[j]))

{

s[j] = tolower(s[j]) + 2;

}

else if(islower(s[j]))

{

s[j] = toupper(s[j]) - 2;

}

else

{

s[j] = '@';

}

cout << "after : " << s[j] << endl;

}

cout << "---------------" << endl;

}

}

int main()

{

char c[] = "Romeo Juliet";

strcon(c); // c = Romeo Juliet

cout << "Text : " << c << endl;

cout << "New Text : " << c + 3 << endl;

cout << "Last Text : " << c + 5 - 2 << endl;

return 0;

}

🡪#include <iostream>

#include <stdio.h>

#define SIZE 2

using namespace std;

struct Faculty

{

int id;

char name[20];

char subject[20];

float salary;

};

void input(struct Faculty &m)

{

cout << "Enter ID : "; cin >> m.id;

cin.ignore();

cout << "Enter name : "; gets(m.name);

cout << "Enter subject : "; gets(m.subject);

cout << "Enter salary : "; cin >> m.salary;

}

void output(struct Faculty m)

{

cout << "ID : " << m.id << endl;

cout << "name : " << m.name << endl;

cout << "subject : " << m.subject << endl;

cout << "salary : " << m.salary << endl;

}

int main()

{

Faculty members[SIZE];

for(int i = 0; i < SIZE; i++)

{

input(members[i]);

}

for(int i = 0; i < SIZE; i++)

{

if(members[i].salary > 10000)

{

output(members[i]);

}

}

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

char string[30];

cout <<"Enter string : "; cin.getline(string, 30);

cout << "string : " << string << endl;

for (int i = 0; string[i] != NULL; i++)

{

if(string[i] == ' ')

{

string[i] = '-';

}

}

cout << "Converted string : " << string << endl;

}

🡪#include <iostream>

using namespace std;

int primefactors(long);

int main()

{

long n;

cin >> n;

long i = 2;

while(n)

{

if(primefactors(i) == 1)

{

// cout << i << endl;

n--;

}

i++;

}

cout << i - 1;

return 0;

}

int primefactors(long n)

{

long i = 2, count = 0;

while(n != 1)

{

if(n % i == 0)

{

count++;

if(count >= 2)

{

return 1;

}

n = n / i;

}

else

{

if(count >= 2)

{

return 1;

}

count = 0;

i++;

}

}

if(count >= 2)

{

return 1;

}

return 0;

}

🡪#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

int stringlen(char \*s)

{

int count = 0;

for(int i = 0; s[i] != NULL; i++)

{

count++;

}

return count;

}

char\* stringConcate(char \*a, char \*b)

{

//this function concatinate two strings

char newstring[50];

int len = 0;

for(int i = 0; a[i] != NULL; i++)

{

newstring[len] = a[i];

len++;

}

for(int i = 0; b[i] != NULL; i++)

{

newstring[len] = b[i];

len++;

}

newstring[len] = NULL;

cout << newstring;

return newstring;

}

int main()

{

// char string1[] = {"kanishk "};

// char string2[] = {"debnath"};

// char sumstring[50];

// strcpy(sumstring, stringConcate(string1, string2));

// cout << "Concatinated string is - " << sumstring << endl;

char string[10];

int n;

cout << "number : ";

cin >> n;

cin.ignore();

cout << "String : ";

gets(string);

cout << endl;

cout << n << endl;

cout << string << endl;

}

🡪#include <iostream>

#include <string.h>

using namespace std;

int main()

{

char s[20];

int flag = 1;

cin.getline(s, 20);

for(int i = 0, l = strlen(s) - 1; i < l; i++, l--)

{

if(s[i] != s[l])

{

flag = 0;

}

}

if(flag == 0)

{

cout << "Not palindrome" << endl;

}

else

{

cout << "palindrome" << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

int stringlen(char\*);

int stringcompare(char\*, char\*);

char \* stringconcatinate(char\*, char\*);

int main()

{

char name[20];

cin.getline(name, 20);

cout << stringlen(name) << endl;

for(int i = 0; i <= stringlen(name); i++)

{

cout.write(name, i);

cout << endl;

}

if(stringcompare("kanishk", "abhinav") == 1)

{

cout << "Same" << endl;

}

else

{

cout << "Different" << endl;

}

char \* sum;

sum = stringconcatinate("kanishk ", "debnath");

cout << sum << endl;

return 0;

}

int stringlen(char \* str)

{

int count = 0;

for (int i = 0; str[i] != NULL; i++)

{

count++;

}

return count;

}

int stringcompare(char\* a, char\* b)

{

int alen = stringlen(a);

int blen = stringlen(b);

if(alen != blen)

{

return 0;

}

for(int i = 0; i < alen; i++)

{

if(a[i] != b[i])

{

return 0;

}

}

return 1;

}

char \* stringconcatinate(char\* a, char\* b)

{

int size = stringlen(a) + stringlen(b);

char \*add;

add = new char[size + 1];

int index = 0;

int i = 0;

int j = 0;

while(i < stringlen(a))

{

add[index] = a[i];

index++;

i++;

}

while(j < stringlen(b))

{

add[index] = b[j];

index++;

j++;

}

add[index] = NULL;

return add;

}

🡪#include <iostream>

#include <stdio.h>

using namespace std;

struct Employee

{

char name[20];

int age;

float salary;

int department[3];

}kunal;

void input(Employee \*emp)

{

cin.ignore();

cout <<"Enter name : "; gets(emp->name);

cout << "Enter age : "; cin >> emp->age;

cout << "Enter salary : "; cin >> emp->salary;

for(int i = 0; i < 3; i++)

{

cout << "Enter department " << i + 1 << " code : ";

cin >> emp->department[i];

}

}

void output(Employee emp)

{

cout << "-----------------------------" << endl;

cout << "name : " << emp.name << endl;

cout << "age : " << emp.age << endl;

cout << "salary : " << emp.salary << endl;

for(int i = 0; i < 3; i++)

{

cout << "department " << i + 1 <<" code : " << emp.department[i] << endl;

}

cout << "-----------------------------" << endl;

}

int main()

{

struct Employee bande[3];

for(int i = 0; i < 3; i++)

{

cout << "banda #"<<i+1<<endl;

input(&bande[i]);

}

for(int i = 0; i < 3; i++)

{

if(bande[i].salary <= 1000)

{

cout << "banda #"<<i+1<<endl;

output(bande[i]);

}

}

input(&kunal);

output(kunal);

}

🡪#include <iostream>

#include <stdio.h>

using namespace std;

struct Student

{

char name[20];

int age;

float marks[3];

float avg;

};

Student input()

{

Student stud;

cin.ignore();

cout << "Enter name : "; gets(stud.name);

cout << "Enter age : "; cin >> stud.age;

for(int j = 0; j < 3; j++)

{

cout << "Enter marks for subject " << j + 1 << " : "; cin >> stud.marks[j];

stud.avg += stud.marks[j];

}

stud.avg /= 3;

return stud;

}

void output(Student std)

{

cout << "Name : " << std.name << endl;

cout << "Age : " << std.age << endl;

for(int j = 0; j < 3; j++)

{

cout << "Marks of subject " << j + 1 << " : " << std.marks[j] << endl;

}

}

int main()

{

Student std[3];

//input processes

for(int i = 0; i < 3; i++)

{

cout << "student #" << i + 1 << endl;

std[i] = input();

}

for(int i = 0; i < 3; i++)

{

if(std[i].avg >= 85.0)

{

output(std[i]);

}

}

}

🡪#include <iostream>

using namespace std;

float powerseries(int, int);

int sumtorial(int);

int power(int, int);

int main()

{

int x, n;

cout << "Enter x : "; cin >> x;

cout << "Enter n : "; cin >> n;

cout << powerseries(n, x);

return 0;

}

float powerseries(int n, int x)

{

float sum = 0;

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

{

cout << "( ";

cout << x << "^" << i << "/ " << sumtorial(i);

cout << ") ";

if(i < n)

{

cout << " + ";

}

sum += (float)power(x, i) / sumtorial(i);

}

cout << " = ";

return sum;

}

int power(int base, int exponent)

{

int pro = 1;

for(int i = 1; i <= exponent; i++)

{

pro \*= base;

}

return pro;

}

int sumtorial(int n)

{

int sum = 0;

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

{

sum += i;

}

return sum;

}

🡪#include <iostream>

using namespace std;

struct Time

{

int hours;

int minutes;

int seconds;

};

//function prototype

struct Time input(int);

void show(struct Time);

int main()

{

struct Time currentTime;

int n;

cout << "Enter time in seconds : "; cin >> n;

currentTime = input(n);

show(currentTime);

return 0;

}

//function definition

struct Time input(int a)

{

struct Time temp;

//logic

temp.hours = a / 3600;

a = a % 3600;

temp.minutes = a / 60;

temp.seconds = a % 60;

return temp;

}

void show(struct Time d)

{

cout << d.hours << " : " << d.minutes << " : " << d.seconds << endl;

}

🡪#include <iostream>

using namespace std;

#define SIZE 3

int main()

{

int matrix[SIZE][SIZE];

int sum = 0;

for(int i = 0; i < SIZE; i++)

{

for(int j = 0; j < SIZE; j++)

{

cin >> matrix[i][j];

}

}

int camelCasing, snake\_case;

for(int i = 0; i < SIZE; i++)

{

for(int j = 0; j < SIZE; j++)

{

if(i != j)

sum += matrix[i][j];

}

}

cout << sum << endl;

return 0;

}

* more

🡪#include <iostream>

#include <math.h>

using namespace std;

int area(int side)

{

return side \* side;

}

int area(int length, int bredth)

{

return length \* bredth;

}

double area(double radius)

{

return 3.14 \* radius \* radius;

}

double area(int a, int b, int c)

{

double s = (a + b + c) / 2;

return sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

int main()

{

//square's area

cout << area(1) << endl;

//rectangle's area

cout << area(12, 5) << endl;

//circle's area

cout << area(1.0) << endl;

//triangle's area

cout << area(3, 4, 5) << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

// returns whether one number divides another or not.

int divide(int, int);

//returns whether the number is prime or not.

int divide(int);

int main()

{

int a, b;

cout << "Enter numbers : " << endl;

cin >> a >> b;

cout << divide(a, b) << endl;

cout << divide(a) << endl;

return 0;

}

int divide(int a, int b)

{

if(a % b == 0)

{

return 1;

}

else

{

return 0;

}

}

int divide(int num)

{

for(int i = 2; i <= num / 2; i++)

{

if(divide(num, i))

{

return 0;

}

}

return 1;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

int countVowel(char \*s)

{

int count = 0;

for(int i = 0; s[i] != NULL; i++)

{

switch(s[i])

{

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

case 'A':

case 'E':

case 'I':

case 'O':

case 'U':count++;

default : continue;

}

}

return count;

}

int main()

{

return 0;

}

🡪#include <iostream>

using namespace std;

int factorial(int n)

{

//base conditions

if(n == 1)

return 1;

// recursion fashion

return n \* factorial(n - 1);

}

int main()

{

int n;

cin >> n;

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

{

cout << factorial(i) << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

int fibonacci(int n)

{

//base conditions

if(n == 1)

return 0;

if(n == 2)

return 1;

// recursion fashion

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main()

{

int n;

cin >> n;

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

{

cout << fibonacci(i) << endl;

}

return 0;

}

🡪#include <iostream>

using namespace std;

void stringReverse(char \*str, int start, int end)

{

// base condition

if(start >= end)

{

return;

}

//logic

char temp;

temp = str[start];

str[start] = str[end];

str[end] = temp;

stringReverse(str, start + 1, end - 1);

}

int main()

{

char name[] = "Stephen Hawkingscl is my favourite teacher.";

int length = sizeof(name) - 1;

cout << length << endl;

cout << "before : " << name << endl;

int start = 0;

int end;

for(int i = 0; name[i] != NULL; i++)

{

if(name[i] == ' ')

{

end = i - 1;

stringReverse(name, start, end);

start = i + 1;

}

}

end = length - 1;

stringReverse(name, start, end);

cout << "after : " << name << endl;

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

long factorial(int x)

{

long product = 1;

for(int i = 1; i <= x; i++)

{

cout << product << " - ";

product = product \* i;

//product \*= i;

}

cout << endl;

return product;

}

long fact(int x)

{

//stopping condition

if(x == 1)

return 1;

//recursive fashion

return x \* fact(x-1);

}

int main()

{

int num;

cout << "Enter the value = ";

cin >> num;

cout << "factorial of " << num << " is = " << fact(num) << endl;

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

// int fibonacci(int n)

// {

// if(n == 1)

// return 0;

//

// if (n == 2)

// return 1;

//

// int sum = 0;

// int first = 0;

// int second = 1;

//

// for(int i = n; i >= 3; i--)

// {

// sum = first + second;

// first = second;

// second = sum;

// }

//

// return sum;

// }

int fibonacci(int n)

{

if(n == 1)

return 0;

if (n == 2)

return 1;

return fibonacci(n - 1) + fibonacci(n - 2);

}

int main()

{

cout << "fibonacci sequence upto 12 terms : " << endl;

for(int i = 1; i <= 12; i++)

{

cout << fibonacci(i) << ", ";

}

cout << endl;

return 0;

}

* Pointers

🡪#include <iostream>

using namespace std;

class Test

{

int \*value;

char \*letter;

public:

Test()

{

value = new int;

letter = new char;

\*value = 0;

\*letter = '\0';

}

Test(int value, char letter)

{

this->value = new int;

this->letter = new char;

this->(\*value) = value;

this->(\*letter) = letter;

}

Test(Test &t)

{

value = t.value;

letter = t.let ter;

}

void get()

{

cout << "Enter value : "; cin >> \*value;

cout << "Enter letter : "; cin >> \*letter;

}

void show()

{

cout << \*value << " ===> " << \*letter << endl;

}

~Test()

{

cout << \*value << " ===> " << \*letter << endl;

cout << "Destructor called! " << endl;

delete value;

delete letter;

}

};

// struct Node

// {

// int value;

// char letter;

// };

//

// void createNodeAndShow(int val, char let)

// {

// Node \*temp;

// temp = new Node;

//

// temp->value = val;

// temp->letter = let;

//

// cout << temp->value << " ==> " << temp->letter << endl;

//

// delete temp;

// }

// void swap(int \*aptr, int \*bptr)

// {

// cout << "inside function : : " << endl;

// cout << "aptr : " << aptr << ", bptr : " << bptr << endl;

//

// int temp;

// temp = \*aptr;

// \*aptr = \*bptr;

// \*bptr = temp;

// cout << "outside function : : " << endl;

// }

int main()

{

//pointer fundamentals

// int x = 20;

// cout << x << endl;

//

// int \*xptr;

// xptr = &x;

// cout << xptr << endl;

//

// int \*\*xpptr;

// xpptr = &xptr;

// cout << xpptr << endl;

//

//

// int a = 5, b = 6;

// int \*aptr = &a, \*bptr = &b;

// cout << "aptr : " << aptr << ", bptr : " << bptr << endl;

//

// cout << "a : " << a << ", b : " << b << endl;

// swap(aptr, bptr);

// cout << "a : " << a << ", b : " << b << endl;

//pointer arithmetics

// int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

// int \*aptr;

// aptr = &array[0];

// cout << aptr << endl;

// for(int i = 0; i < 10; i++)

// {

// cout << \*aptr << " ===> " << aptr << endl;

// aptr++;

// }

//dynamic allocation

// int n;

// cin >> n;

// int array[n];

// cout << "Enter elements : ";

//

//

// for (int i = 0; i < n; i++)

// {

// cin >> array[i];

// }

//

// for (int i = 0; i < n; i++)

// {

// cout << array[i] << endl;

// }

// int size;

// int \*array;

// cout << "Enter size of the array : "; cin >> size;

// array = new int[size];

// for (int i = 0; i < size-1; i++)

// {

// cin >> array[i];

// }

//

// for (int i = 0; i < size; i++)

// {

// cout << array[i] << endl;

// }

//

// delete [] array;

//dynamic structures and classes

// Node node;

//

// cout << "enter value : ";

// cin >> node.value;

// cin.ignore();

// cout << "enter letter : ";

// cin >> node.letter;

//

// cout << node.value << " ==> " << node.letter << endl;

// createNodeAndShow(154, 'a');

Test t(54, 'j');

//this pointer

return 0;

}

🡪#include <iostream>

using namespace std;

int main()

{

int arr[] = {10, 23, 30, 40};

int \*ptr = arr;

int val = \*ptr;

cout << val << endl;

val = \*ptr++;

cout << val << endl;

val = \*ptr;

cout << val << endl;

val = \*++ptr;

cout << val << endl;

return 0;

}

* Classes,constructors and inheritance

#include <iostream>

using namespace std;

//class definition

class Complex

{

//private members

private:

int real;

int imaginary;

//public:

public:

//constructors

//default constructors

// Complex()

// {

// real = 10;

// imaginary = 40;

// }

//parameterised constructors

Complex(int r = -1, int i = -1)

{

real = r;

imaginary = i;

}

//copy constructors

Complex(Complex &s)

{

real = s.imaginary;

imaginary = s.real;

}

Complex(Complex &s, Complex &k)

{

real = s.real + k.real;

imaginary = s.imaginary + k.imaginary;

}

//destructor

~Complex()

{

cout << "---------------------------------" << endl;

cout << real << " i" << imaginary << endl;

cout << "destructor called saksham" << endl;

cout << "---------------------------------" << endl;

}

void set(int r, int i)

{

real = r;

imaginary = i;

}

void get()

{

cout << real << " + i(" << imaginary << ")";

}

};

int main()

{

Complex c, d;

Complex e, f(45, 100);

Complex g(f);

Complex h = c;

Complex k(f, g);

c.set(3, 5);

d.set(5, -56);

c.get();

cout << endl;

d.get();

cout << endl;

e.get();

cout << endl;

f.get();

cout << endl;

g.get();

cout << endl;

h.get();

cout << endl;

k.get();

cout << endl;

return 0;

}

🡪 #include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

struct Time

{

int arrivalTime;

int departureTime;

};

class Employee

{

private:

int eno;

char name[20];

float salary;

Time workTime;

public:

//default constructor

Employee()

{

eno = 0;

strcpy(name, "undefined");

salary = 0.0;

workTime.arrivalTime = 0;

workTime.departureTime = 0;

}

//parameterized constructor

Employee(int num, char \* naam, float sal, int arrival, int departure)

{

eno = num;

strcpy(name, naam);

salary = sal;

workTime.arrivalTime = arrival;

workTime.departureTime = departure;

}

//copy constructor

Employee(Employee &employee)

{

eno = employee.eno;

strcpy(name, employee.name);

salary = employee.salary;

workTime.arrivalTime = employee.workTime.arrivalTime;

workTime.departureTime = employee.workTime.departureTime;

}

~Employee()

{

cout << "destructor for employee " << eno << " called." << endl;

}

void get\_details();

void show\_details();

};

void promotedemployee(Employee \*);

int main()

{

return 0;

}

void Employee::get\_details()

{

cout << "Enter employee number : "; cin >> eno;

cin.ignore();

cout << "Enter name : "; gets(name);

cout << "Enter salary : "; cin >> salary;

cout << "Enter arrival time and departure time : ";

cin >> workTime.arrivalTime >> workTime.departureTime;

}

void Employee::show\_details()

{

cout << "---------------------------------------------------------------" << endl;

cout << "Employee number : " << eno << endl;

cout << "Employee name : " << name << endl;

cout << "Salary : " << salary << endl;

cout << "Work time : " << workTime.arrivalTime << " - " << workTime.departureTime << endl;

cout << "---------------------------------------------------------------" << endl;

}

void promotedemployee(Employee \*employee)

{

//loop through all employees

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

{

//if salary < 100000 and work hours > 7

if(employee[i].salary < 100000 && employee[i].workingHours > 7)

{

//update salary = salary \* 1.1

employee[i].updateSalary();

//display details

}

🡪 #include <iostream>

using namespace std;

class Resort

{

private:

int rno;

char name[20];

float charges;

int days;

float compute();

public:

void getinfo();

void dispinfo();

};

float Resort::compute()

{

return days \* charges;

}

void Resort::getinfo()

{

//char buffer[2];

cout << "Enter room number : ";

cin >> rno;

//cin.getline(buffer, 2);

cin.ignore();

cout << "Enter name : ";

cin.getline(name, 20);

cout << "Enter number of days : ";

cin >> days;

cout << "Enter charges per day : ";

cin >> charges;

}

void Resort::dispinfo()

{

cout << "Room number : " << rno << endl;

cout << "Name : " << name << endl;

cout << "Days : " << days << endl;

cout << "Charges per day : " << charges << endl;

cout << "Total amount : " << compute() << endl;

}

int main()

{

Resort r;

r.getinfo();

r.dispinfo();

return 0;

}

🡪#include <iostream>

#include <string>

using namespace std;

int main()

{

string name;

name = "kanishk debnath";

cout << name << endl;

if(name.find("kan") != std::string::npos)

{

cout << "aaye haaye!" << endl;

}

return 0;

}

* #include <iostream>

using namespace std;

class Student

{

private:

int rollno;

char name[20];

float marks[5];

char grade();

public:

void get\_data();

void show\_data();

};

char Student::grade()

{

float avg = 0.0;

for(int i = 0; i < 5 ; i++)

{

avg += marks[i];

}

avg /= 5.0;

if(avg <= 30)

return 'F';

else if(avg <= 50)

return 'E';

else if(avg <= 60)

return 'D';

else if(avg <= 70)

return 'C';

else if(avg <= 80)

return 'B';

else if(avg <= 90)

return 'A';

else if(avg <= 100)

return 'O';

else

return '~';

}

void Student::get\_data()

{

cout << "Enter roll number : "; cin >> rollno;

cin.ignore();

cout << "Enter name : "; cin.getline(name, 20);

cout << "Enter marks : " << endl;

for(int i = 0 ; i < 5; i++)

{

cout << "marks " << i + 1 << " : ";

cin >> marks[i];

}

}

void Student::show\_data()

{

cout << "Roll number : " << rollno << endl;

cout << "Name : " << name << endl;

cout << "Grade : " << grade() << endl;

}

int main()

{

Student student[5];

for(int i = 0; i < 5; i++)

{

student[i].get\_data();

}

for(int i = 0; i < 5; i++)

{

student[i].show\_data();

}

return 0;

}

🡪 #include <iostream>

using namespace std;

class Time

{

private:

int hr, min, sec;

public:

Time()

{

hr=5;

min=8;

sec=6;

}

Time(int h, int m, int s)

{

hr=h;

min=m;

sec=s;

}

Time(Time &d)

{

hr=d.hr;

min=d.min;

sec=d.sec;

}

void set\_time();

void get\_time();

};

void Time::set\_time()

{

cout<<"enter hour"<<endl;

cin>>hr;

cout<<"enter minutes"<<endl;

cin>>min;

cout<<"enter seconds"<<endl;

cin>>sec;

}

void Time::get\_time()

{

cout<<hr<<":"<<min<<":"<<sec;

}

int main()

{

Time a, b;

Time e=a;

Time c(3, 5, 8);

//a.set\_time();

b.set\_time();

Time g(b);

a.get\_time();

cout<<endl;

b.get\_time();

cout<<endl;

c.get\_time();

cout<<endl;

g.get\_time();

cout<<endl;

e.get\_time();

return 0;

}

🡪 #include <iostream>

using namespace std;

class Train

{

private:

//data members

int trainNo; //train number

int distance; //distance travelled by the train

int fuel; //amount of fuel required at max

float amount; //amount of ticket

//private member functions

void settleAmount(); //calculate the amount of ticket

void settleFuel(); //calculate the fuel required

public:

//member functions

void getTicket();

void showTicket();

};

//function definitions

void Train::getTicket()

{

cout << "Enter train number : "; cin >> trainNo;

cout << "Enter distance : "; cin >> distance;

settleFuel();

settleAmount();

}

void Train::settleFuel()

{

if(distance <= 1500)

{

fuel = 250;

}

else if(distance > 1500 && distance <= 3000)

{

fuel = 1000;

}

else

{

fuel = 2500;

}

}

void Train::showTicket()

{

cout << "Train Number : " << trainNo << endl;

cout << "Distance : " << distance << endl;

cout << "Amount : " << amount << endl;

cout << "Fuel capacity : " << fuel << endl;

}

void Train::settleAmount()

{

if(distance <= 1000)

{

cout << "here";

amount = float(distance \* 0.5);

}

else if(distance > 1000 && distance <= 2000)

{

amount = float(distance \* 0.4);

}

else if(distance > 2000 && distance <= 3000)

{

amount = float(distance \* 0.3);

}

else

{

amount = float(distance \* 0.2);

}

}

int main()

{

Train t;

t.getTicket();

t.showTicket();

return 0; }

🡪#include <iostream>

using namespace std;

class Time

{

int seconds;

int minutes;

int hours;

public:

Time()

{

seconds = 0;

minutes = 0;

hours = 0;

}

void input();

void output();

};

class Date

{

int day;

int month;

int year;

public:

Date()

{

day = 0;

month = 0;

year = 0;

}

void input();

void output();

};

void Time::input()

{

cout << "(hh:mm:ss) : ";

cin >> hours >> minutes >> seconds;

}

void Date::input()

{

cout << "(dd:mm:yy) : ";

cin >> day >> month >> year;

}

void Time::output()

{

cout << hours << " : " << minutes << " : " << seconds;

}

void Date::output()

{

cout << day << " : " << month << " : " << year;

}

class Player

{

//using containership/aggregation

char name[20];

int age;

Time training\_time;

Date dob;

Date doj;

public:

void get();

void put();

};

void Player::get()

{

cin.ignore();

cout << "Enter name : "; cin.getline(name, 20);

cout << "Enter age : "; cin >> age;

cout << "Enter training time : "; training\_time.input();

cout << "Enter DOB : "; dob.input();

cout << "Enter DOJ : "; doj.input();

}

void Player::put()

{

cout << "Name : " << name << endl;

cout << "Age : " << age << endl;

cout << "Training time : "; training\_time.output(); cout << endl;

cout << "DOB : "; dob.output(); cout << endl;

cout << "DOJ : "; doj.output(); cout << endl;

}

int main()

{

Player player;

cout << "Class called ! " << endl;

player.get();

player.put();

return 0;

}

🡪#include <iostream>

using namespace std;

////////////////////////////////////////

class Distance

{

private:

int km;

int m;

public:

//constructors

Distance(); //default constructor

Distance(int, int); //parameterised constructor

Distance(Distance &); //copy constructor

//destructor

~Distance();

//prototype

void put\_distance(void);

void get\_distance();

};

//defintions

Distance::Distance()

{

km = 0;

m = 0;

//put\_distance();

}

Distance::Distance(int a, int b = 1000)

{

km = a;

m = b;

}

Distance::Distance(Distance &t)

{

km = t.km;

m = t.m;

}

Distance::~Distance()

{

//get\_distance();

cout << "Destructor called " << km << " == " << m << endl;

}

void Distance::put\_distance(void)

{

cout << "put function called" << endl;

cout << "Enter distance : ";

cout << "km = "; cin >> km;

cout << "m = "; cin >> m;

}

void Distance::get\_distance()

{

cout << km << " km " << m << " m." << endl;

}

////////////////////////////////////////

int main()

{

Distance d1;

Distance d2;

cout << "d2 : ";

d2.get\_distance();

d2 = Distance(450, 123);

Distance d3(56, 78);

Distance d4(d2);

Distance d5(340);

Distance d6;

// d.put\_distance();

d1.get\_distance();

d2.get\_distance();

d3.get\_distance();

d4.get\_distance();

d5.get\_distance();

d6.get\_distance();

return 0;

}

🡪#include <iostream>

using namespace std;

class Test

{

int a;

char b;

static int count;

public:

Test();

Test(int, char);

~Test();

static void showCount()

{

cout << "Count = " << count << endl;

}

};

int Test::count = 0;

Test::Test()

{

//default constructor

a = 0;

b = 'k';

cout << a << " ---> " << b << endl;

cout << "default constructor called." << endl;

count++;

}

Test::Test(int num, char ch)

{

a = num;

b = ch;

cout << a << " ---> " << b << endl;

cout << "parameterised constructor called." << endl;

count++;

}

Test::~Test()

{

cout << a << " ---> " << b << endl;

cout << "Destructor called." << endl;

count--;

}

int main()

{

Test obj; //default constructor called

obj.showCount();

Test obj2(12, 'a'); //implicit called parameterised constructor

obj2.showCount();

Test obj3;

obj3.showCount();

obj3 = Test(65, 'e'); //explicit called parameterised constructor

obj3.showCount();

Test obj4 = Test(56, 'g');

obj4.showCount();

return 0;

}

🡪#include <iostream>

using namespace std;

class Base

{

int a, b;

public:

void get();

void show();

};

class Derived : public Base

{

int c, d;

public:

// void insert();

// void display();

void fun()

{

cout << "jai mata di" << endl;

}

};

void Base::get()

{

cout << "Base get function called !" << endl;

cout << "a : "; cin >> a;

cout << "b : "; cin >> b;

}

void Base::show()

{

cout << "Base show function called !" << endl;

cout << "a : " << a << endl;

cout << "b : " << b << endl;

}

// void Derived::insert()

// {

// cout << "Derived insert function called !" << endl;

// cout << "c : "; cin >> c;

// cout << "d : "; cin >> d;

// }

//

// void Derived::display()

// {

// cout << "Derived display function called !" << endl;

// cout << "c : " << c << endl;

// cout << "d : " << d << endl;

// }

int main()

{

Base b;

Derived d;

b.get();

b.show();

d.get();

d.show();

d.fun();

return 0;

}

🡪#include <iostream>

using namespace std;

class Base

{

int a, b;

public:

void get();

void show();

};

class Derived : public Base

{

int c, d;

public:

// void insert();

// void display();

void fun()

{

cout << "jai mata di" << endl;

}

};

void Base::get()

{

cout << "Base get function called !" << endl;

cout << "a : "; cin >> a;

cout << "b : "; cin >> b;

}

void Base::show()

{

cout << "Base show function called !" << endl;

cout << "a : " << a << endl;

cout << "b : " << b << endl;

}

// void Derived::insert()

// {

// cout << "Derived insert function called !" << endl;

// cout << "c : "; cin >> c;

// cout << "d : "; cin >> d;

// }

//

// void Derived::display()

// {

// cout << "Derived display function called !" << endl;

// cout << "c : " << c << endl;

// cout << "d : " << d << endl;

// }

int main()

{

Base b;

Derived d;

b.get();

b.show();

d.get();

d.show();

d.fun();

return 0;

}

🡪#include <iostream>

using namespace std;

class School

{

protected:

char name[20];

int age;

char address[50];

public:

//home assignment

void get\_common\_data();

void show\_common\_data();

};

class Student : public School

{

int standard;

int roll\_number;

public:

void get\_details();

void show\_details();

};

void Student::get\_details()

{

cout << "Enter roll number : "; cin >> roll\_number;

cin.ignore();

cout << "Enter name : "; cin.getline(name, 20);

cout << "Enter age : "; cin >> age;

cout << "Enter standard : "; cin >> standard;

cin.ignore();

cout << "Enter address : "; cin.getline(address, 50);

}

void Student::show\_details()

{

cout << "Roll number : " << roll\_number << endl;

cout << "Name : " << name << endl;

cout << "Standard : " << standard << endl;

cout << "Age : " << age << endl;

cout << "Address : " << address << endl;

}

int main()

{

Student student;

student.get\_details();

student.show\_details();

return 0;

}

🡪#include <iostream>

using namespace std;

class Test

{

int a;

char b;

static int count;

public:

Test();

Test(int, char);

~Test();

static void showCount()

{

cout << "Count = " << count << endl;

}

};

int Test::count = 0;

Test::Test()

{

//default constructor

a = 0;

b = 'k';

cout << a << " ---> " << b << endl;

cout << "default constructor called." << endl;

count++;

}

Test::Test(int num, char ch)

{

a = num;

b = ch;

cout << a << " ---> " << b << endl;

cout << "parameterised constructor called." << endl;

count++;

}

Test::~Test()

{

cout << a << " ---> " << b << endl;

cout << "Destructor called." << endl;

count--;

}

int main()

{

Test obj; //default constructor called

obj.showCount();

Test obj2(12, 'a'); //implicit called parameterised constructor

obj2.showCount();

Test obj3;

obj3.showCount();

obj3 = Test(65, 'e'); //explicit called parameterised constructor

obj3.showCount();

Test obj4 = Test(56, 'g');

obj4.showCount();

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

struct Date

{

int day;

int month;

int year;

};

class Product

{

private:

float price;

int quantity;

char id[5];

char name[40];

Date manufacture\_date;

Date expiry\_date;

float total\_cost()

{

return price \* quantity;

}

public:

Product()

{

//default constructor

price = 10;

quantity = 10;

strcpy(id, "0000");

strcpy(name, "none");

manufacture\_date.day = 10;

manufacture\_date.month = 10;

manufacture\_date.year = 10;

expiry\_date.day = 10;

expiry\_date.month = 10;

expiry\_date.year = 10;

}

Product(float p, char i[5], char n[40])

{

//parameterised constructor

price = p;

strcpy(id, i);

strcpy(name, n);

quantity = 10;

manufacture\_date.day = 10;

manufacture\_date.month = 10;

manufacture\_date.year = 10;

expiry\_date.day = 10;

expiry\_date.month = 10;

expiry\_date.year = 10;

}

Product(Product &object)

{

//copy constructor

price = object.price;

quantity = object.quantity;

strcpy(id, object.id);

strcpy(name, object.name);

manufacture\_date.day = object.manufacture\_date.day;

manufacture\_date.month = object.manufacture\_date.month;

manufacture\_date.year = object.manufacture\_date.year;

expiry\_date.day = object.expiry\_date.day;

expiry\_date.month = object.expiry\_date.month;

expiry\_date.year = object.expiry\_date.year;

}

void get();

void show()

{

cout << "serial ID : ";

cout << id;

cout << endl;

cout << "name : ";

cout << name;

cout << endl;

cout << "quantity : ";

cout << quantity;

cout << endl;

cout << "price : ";

cout << price;

cout << endl;

cout << "date of maufacturing : ";

cout << manufacture\_date.day << "/" << manufacture\_date.month << "/" << manufacture\_date.year;

cout << endl;

cout << "date of expiry : ";

cout << expiry\_date.day << "/" << expiry\_date.month << "/" << expiry\_date.year;

cout << endl;

cout << "Total cost = " << total\_cost() << endl;

}

void quality()

{

if(expiry\_date.year - manufacture\_date.year > 3)

{

cout << "stale" << endl;

}

else if(expiry\_date.year - manufacture\_date.year < 3)

{

cout << "fresh" << endl;

}

else

{

if(expiry\_date.month > manufacture\_date.month)

{

cout << "fresh" << endl;

}

else if(expiry\_date.month < manufacture\_date.month)

{

cout << "stale" << endl;

}

else

{

if(expiry\_date.day >= manufacture\_date.day)

{

cout << "fresh" << endl;

}

else

{

cout << "stale" << endl;

}

}

}

}

~Product()

{

cout << name << " is destructed. " << endl;

}

};

void Product::get()

{

cout << "Enter serial ID : ";

cin.getline(id, sizeof(id));

cin.ignore();

cout << "Enter name : ";

cin.getline(name, sizeof(name));

cout << "Enter quantity : ";

cin >> quantity;

cout << "Enter price : ";

cin >> price;

cout << "Enter date of maufacturing : " << endl;

cin >> manufacture\_date.day >> manufacture\_date.month >> manufacture\_date.year;

cout << "Enter date of expiry : " << endl;

cin >> expiry\_date.day >> expiry\_date.month >> expiry\_date.year;

}

int main()

{

Product product[3];

Product product\_1(120, "1200", "zero");

Product product\_2(product\_1);

product.show();

product\_1.show();

product\_2.show();

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

class Bank

{

char name[20];

char acctype;

int accno;

float balance;

public:

Bank()

{

accno = 7;

strcpy(name, "mainstream tushar");

acctype = 's';

balance = 0;

}

Bank(int accno, char name[20], char acctype)

{

this->accno = accno;

strcpy(this->name, name);

this->acctype = acctype;

this->balance = 0;

}

Bank(Bank &object)

{

this->accno = object.accno;

strcpy(this->name, object.name);

this->acctype = object.acctype;

this->balance = 0;

}

void initialise()

{

cin >> accno;

cin.ignore();

cin.getline(name, 20);

cin >> acctype;

}

void deposit()

{

float amt;

cin >> amt;

balance += amt;

}

void withdraw()

{

float amt;

cin >> amt;

if(amt <= balance)

{

balance -= amt;

}

else

{

cout << "Insufficient balance." << endl;

}

}

void display()

{

cout << accno << endl;

cout << name << endl;

cout << acctype << endl;

cout << balance << endl;

}

};

int main()

{

Bank bank1;

Bank bank2(150, "kanishk", 'S');

Bank bank3(bank2);

Bank bank4;

bank4 = bank1;

bank1.display();

bank2.display();

bank3.display();

bank4.display();

return 0;

}

🡪#include <iostream>

using namespace std;

class Test

{

int a, b, c;

static int count;

public:

Test(int k, int l, int h)

{

a = k;

b = l;

c = h;

}

void show()

{

cout << a << " " << b << " " << c << endl;

}

static void showCount()

{

count++;

cout << count << endl;

}

};

int Test::count;

int main()

{

Test t1(123, 32, 546);

t1.show();

t1.showCount();

Test t2(45, 546, 68);

t2.show();

t2.showCount();

Test t3(45, 546, 68);

t2.show();

t2.showCount();

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

class Student

{

char name[20];

int roll;

float marks[5]; //array within a class

float average() const; //mutator functions

static int count;

public:

//manager functions

Student()

{

strcpy(name, "None");

roll = 0;

for(int i = 0; i < 5; i++)

marks[i] = 0;

}

Student(char name[20], int roll, float marks[5], int day, int month, int year)

{

strcpy(this->name, name);

this->roll = roll;

for(int i = 0; i < 5; i++)

this->marks[i] = marks[i];

}

Student(Student &object)

{

strcpy(this->name, object.name);

this->roll = object.roll;

for(int i = 0; i < 5; i++)

this->marks[i] = object.marks[i];

}

void get(); //accessor functions

void show();

void showCount()

{

cout << "Count is " << count << endl;

}

};

int Student::count = 0;

float Student::average() const

{

float sum = 0;

for(int i = 0; i < 5; i++)

sum += marks[i];

return sum / 5;

}

void Student::get()

{

cout << "Enter roll number : ";

cin >> roll;

cin.ignore();

cout << "Enter name : ";

cin.getline(name, 20);

cout << "Enter marks for : " << endl;

for (int i = 0; i < 5; i++)

{

cout << "Subject " << i + 1 << " : ";

cin >> marks[i];

}

count++;

}

void Student::show()

{

cout << "roll number : ";

cout << roll;

cout << endl;

cout << "name : ";

cout << name;

cout << endl;

cout << "marks for : " << endl;

for (int i = 0; i < 5; i++)

{

cout << "Subject " << i + 1 << " : ";

cout << marks[i];

cout << endl;

}

cout << "Average : " << average() << endl;

}

void hoolalala(Student student[], int n)

{

cout << "details of students : " << endl;

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

{

student[i].show();

}

}

int main()

{

int n;

cout << "Enter number of students : ";

cin >> n;

Student student[n];

cout << "Enter details of students : " << endl;

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

{

student[i].get();

student[i].showCount();

}

hoolalala(student, n);

return 0;

}

* File handling

🡪🡪//template for line to line or character to character type questions of file handling

#include <iostream>

#include <fstream>

using namespace std;

void something(char \* line)

{

//question's logic

}

int main()

{

char line[100];

//reading part

ifstream file;

file.open("file.txt", ios::in);

while(file.getline(line, sizeof(line)))

{

//logic code

something(word);

}

//remainder section of code

file.close();

return 0;

}

🡪//template for word to word type questions of file handling

#include <iostream>

#include <fstream>

using namespace std;

void something(char \* word)

{

//question's logic

}

int main()

{

char word[20];

//reading part

ifstream file;

file.open("file.txt", ios::in);

while(file >> word)

{

//logic code

something(word);

}

//remainder section of code

file.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

int main()

{

int a;

cout << "Enter any integer : ";

cin >> a;

//inserted item to the file

ofstream fout;

fout.open("test1.txt", ios::out);

fout << a;

fout.close();

//getting item from the file

ifstream fin;

fin.open("test1.txt", ios::in);

fin >> a;

fin.close();

cout << a;

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

int main()

{

char a[50];

int count = 0;

cout << "Enter any string : ";

cin.getline(a, 50);

//inserted item to the file

ofstream fout;

fout.open("test2.txt", ios::out);

fout << a;

fout.close();

//getting item from the file

ifstream fin;

fin.open("test2.txt", ios::in);

while(!fin.eof())

{

fin.getline(a, 50);

cout << a << endl;

// count++;

}

fin.close();

// cout << count - 2 << endl;

return 0;

}

🡪#include <iostream>

#include <fstream>

#include <ctype.h>

using namespace std;

int upperCount(char \* s)

{

int count = 0;

for(int i = 0; s[i] != NULL; i++)

{

if(isupper(s[i]))

{

count++;

}

}

return count;

}

int lowerCount(char \* s)

{

int count = 0;

for(int i = 0; s[i] != NULL; i++)

{

if(islower(s[i]))

{

count++;

}

}

return count;

}

int main()

{

char s[100];

cout << "Enter your string : ";

cin.getline(s, 100);

ofstream fout;

fout.open("test3.txt", ios::out | ios::nocreate);

fout << s;

// ifstream fin;

// fin.open("test3.txt", ios::in);

// fin.getline(s, 100);

//

// cout << s << endl;

// cout << "Cap count : " << upperCount(s) << endl;

// cout << "Lower count : " << lowerCount(s) << endl;

//

// fin.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

struct Employee

{

int eno;

char name[20];

float salary;

};

int main()

{

int num;

Employee e[10];

// cout << "Enter the number of details you want to enter : ";

// cin >> num;

//

// for(int i = 0; i < num; i++)

// {

// cout << "Enter eno : "; cin >> e[i].eno;

// cin.ignore();

// cout << "Enter name : "; cin.getline(e[i].name, 20);

// cout << "Enter salary : "; cin >> e[i].salary;

// }

// fstream fout;

//

// fout.open("test4.txt", ios::out);

// // file.write((char \*) & object, sizeof(object));

//

// for(int i = 0; i < num; i++)

// {

// fout.write((char \*) & e[i], sizeof(e[i]));

// }

//

// fout.close();

ifstream fin;

fin.open("test4.txt", ios::in);

for(int i = 0; i < 3; i++)

{

fin.read((char \*) & e[i], sizeof(e[i]));

cout << e[i].eno << " - " << e[i].name << " - " << e[i].salary << endl;

}

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll;

char name[20];

public:

void get()

{

cout << "Enter roll number : ";

cin >> roll;

cin.ignore();

cout << "Enter the name : ";

cin.getline(name, 20);

}

void show()

{

cout << "Roll number : " << roll << endl;

cout << "Name : " << name << endl;

}

};

int main()

{

Student s[3];

for(int i = 0; i < 3; i++)

{

cout << "Student " << i + 1 << endl;

s[i].get();

}

ofstream fout;

fout.open("test5.txt", ios::out);

for(int i = 0; i < 3; i++)

{

fout.write((char \*) &s[i], sizeof(s[i]));

}

fout.close();

//reading objects from file

// ifstream fin;

// fin.open("test5.txt", ios::in);

//

// for(int i = 0; i < 3; i++)

// {

// fin.read((char \*) &s[i], sizeof(s[i]));

// cout << "Student " << i + 1 << endl;

// s[i].show();

// }

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll;

char name[20];

public:

void get()

{

cout << "Enter roll number : ";

cin >> roll;

cin.ignore();

cout << "Enter the name : ";

cin.getline(name, 20);

}

void show()

{

cout << "Roll number : " << roll << endl;

cout << "Name : " << name << endl;

}

};

class Marks

{

int subject;

char remark[20];

public:

void get()

{

cout << "Enter Subject : ";

cin >> subject;

cin.ignore();

cout << "Enter the Remark : ";

cin.getline(remark, 20);

}

void show()

{

cout << "Subject : " << subject << endl;

cout << "Remark : " << remark << endl;

}

};

int main()

{

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll;

char name[20];

public:

void get()

{

cout << "Enter roll number : ";

cin >> roll;

cin.ignore();

cout << "Enter the name : ";

cin.getline(name, 20);

}

void show()

{

cout << "Roll number : " << roll << endl;

cout << "Name : " << name << endl;

}

int get\_roll()

{

return roll;

}

};

int main()

{

// Student s[3];

// for(int i = 0; i < 3; i++)

// {

// cout << "Student " << i + 1 << endl;

// s[i].get();

// }

//

// ofstream fout;

// fout.open("sortedStud.txt", ios::out);

// for(int i = 0; i < 3; i++)

// {

// fout.write((char \*) &s[i], sizeof(s[i]));

// }

//

// fout.close();

//reading objects from file

// ifstream fin;

// fin.open("test5.txt", ios::in);

//

// for(int i = 0; i < 3; i++)

// {

// fin.read((char \*) &s[i], sizeof(s[i]));

// cout << "Student " << i + 1 << endl;

// s[i].show();

// }

//main input

Student input;

Student s;

input.get();

ofstream tempfile;

tempfile.open("temp.txt", ios::out); //writing file

ifstream mainfile;

mainfile.open("sortedStud.txt", ios::in); //reading file

while(!mainfile.eof())

{

mainfile.read( (char \*) &s, sizeof(s));

if(s.get\_roll() < input.get\_roll())

{

tempfile.write((char \*) &s, sizeof(s));

}

else

{

tempfile.write((char \*) &input, sizeof(input));

break;

}

}

while(!mainfile.eof())

{

mainfile.read( (char \*) &s, sizeof(s));

tempfile.write((char \*) &s, sizeof(s));

}

remove("sortedStud.txt");

rename("temp.txt", "sortedStud.txt");

mainfile.close();

tempfile.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

#include <ctype.h>

using namespace std;

int countDigit(char \*line)

{

int count = 0;

for(int i = 0; line[i] != NULL; i++)

{

if(isdigit(line[i]))

count++;

}

return count;

}

int main()

{

char input[100];

cout << "Enter your text : " << endl;

cin.getline(input, sizeof(input));

ofstream fout;

fout.open("text.txt", ios::out);

fout << input;

fout.close();

ifstream fin;

fin.open("text.txt", ios::in);

char line[100];

int totalDigits = 0;

int lineCounter = 0;

int cd = 0;

while (fin.getline(line, sizeof(line)))

{

cd = countDigit(line);

totalDigits += cd;

cout << ++lineCounter << " : " << cd << endl;

}

cout << "Total number of digits = " << totalDigits << endl;

fin.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll;

char name[20];

public:

void get()

{

cout << "Enter roll number : ";

cin >> roll;

cin.ignore();

cout << "Enter the name : ";

cin.getline(name, 20);

}

void show()

{

cout << "Roll number : " << roll << endl;

cout << "Name : " << name << endl;

}

int get\_roll()

{

return roll;

}

};

int main()

{

// Student s[3];

//

// for(int i = 0; i < 3; i++)

// {

// cout << "Student " << i + 1 << endl;

// s[i].get();

// }

//

// ofstream fout;

// fout.open("student.dat", ios::out | ios::binary);

// for(int i = 0; i < 3; i++)

// {

// fout.write((char \*) &s[i], sizeof(s[i]));

// }

//

// fout.close();

//deletion of record

ifstream mainfile;

mainfile.open("student.dat", ios::in | ios::binary);

if(!mainfile)

{

cout << "file doesn't exist." << endl;

return 0;

}

Student s;

int roll\_number;

cout << "Enter the roll number to be deleted : ";

cin >> roll\_number;

ofstream tempfile;

tempfile.open("temp.dat", ios::out | ios::binary);

while(!mainfile.eof())

{

mainfile.read((char\*) &s, sizeof(s));

if(roll\_number != s.get\_roll())

{

tempfile.write((char \*) &s, sizeof(s));

}

else

{

continue;

}

}

remove("student.dat");

rename("temp.dat", "student.dat");

tempfile.close();

mainfile.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

void hisToHer(char \*line)

{

if((line[0] == 'h' || line[0] == 'H') && (line[1] == 'i' || line[1] == 'I') && (line[2] == 's' || line[2] == 'S'))

{

line[1] = 'e';

line[2] = 'r';

}

for(int i = 3; line[i] != NULL; i++)

{

if(line[i] == ' ')

{

if((line[i+1] == 'h' || line[i+1] == 'H') && (line[i+2] == 'i' || line[i+2] == 'I') && (line[i+3] == 's' || line[i+3] == 'S'))

{

line[i+2] = 'e';

line[i+3] = 'r';

}

}

}

}

int countThreeLetterWords(char \*line)

{

int lastSpaceIndex = 0;

int count = 0;

if(line[3] == ' ')

{

count++;

lastSpaceIndex = 3;

}

for(int i = 4; line[i] != NULL; i++)

{

if(line[i] == ' ')

{

if((i - lastSpaceIndex) == 4)

{

count++;

}

lastSpaceIndex = i;

}

}

return count;

}

int main()

{

ifstream fin;

fin.open("notes.txt", ios::in);

char text[100];

fin.getline(text, sizeof(text));

hisToHer(text);

cout << text << endl;

cout << countThreeLetterWords(text) << endl;

fin.close();

ofstream fout;

fout.open("notes.txt", ios::out);

fout << text;

fout.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

#include <ctype.h>

using namespace std;

int countVowel(char \*s)

{

int count = 0;

for(int i = 0; s[i] != NULL; i++)

{

switch(s[i])

{

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

case 'A':

case 'E':

case 'I':

case 'O':

case 'U':count++;

default : continue;

}

}

return count;

}

int countUpper(char \*s)

{

int count = 0;

for(int i = 0; s[i] != NULL; i++)

{

if(isupper(s[i]))

count++;

}

return count;

}

int countWord(char \*s)

{

int count = 1;

for(int i = 0; s[i] != NULL; i++)

{

if(s[i] == ' ')

count++;

}

return count;

}

int countSpecial(char \*s)

{

int count = 0;

for(int i = 0; s[i] != '\0'; i++)

{

if(!isalpha(s[i]) && s[i] != ' ')

count++;

}

return count;

}

int main()

{

ifstream file;

file.open("poem.txt", ios::in);

int lineCounter = 0;

char line[100];

int totalVowels = 0;

while(file.getline(line, 100))

{

cout << line << endl;

cout << "Uppers in line " << ++lineCounter << " -> " << countWord(line) << endl;

totalVowels += countVowel(line);

}

cout << lineCounter << endl;

cout << totalVowels << endl;

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

int isAorI(char \* line)

{

if(line[0] == 'a' || line[0] == 'A')

return 1;

if(line[0] == 'i' || line[0] == 'I')

return 1;

return 0;

}

int main()

{

int count = 0;

ifstream fin;

fin.open("test.txt", ios::in);

char line[100];

int i = 1;

while(fin.getline(line, sizeof(line)))

{

if(isAorI(line) == 1)

{

cout << i << " - " << line << endl;

count++;

}

i++;

}

cout << endl << "Count is = " << count << endl;

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

#include <fstream>

using namespace std;

//word to word

int isThe(char \* word)

{

if(strcmp(word, "the") == 0 || strcmp(word, "The") == 0)

{

return 1;

}

return 0;

}

isD(char \* word)

{

int length = strlen(word);

if(length % 2 == 0)

return 0;

int mid = length / 2;

if(word[mid] == 'd' || word[mid] == 'D')

{

return 1;

}

return 0;

}

isG(char \* word)

{

int length = strlen(word);

if(length % 2 == 0)

return 0;

int mid = length / 2;

if(word[mid] == 'g' || word[mid] == 'G')

{

return 1;

}

return 0;

}

int main()

{

int count = 0;

ifstream fin;

fin.open("test.txt", ios::in);

char word[20];

int i = 0;

while(fin >> word)

{

if(isG(word) == 1)

{

cout << i << " - ";

cout << word << endl;

count++;

}

i++;

}

cout << endl << "Count of the is = " << count << endl;

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Test

{

int a;

char name[20];

public:

void get()

{

cout << "Enter a : ";

cin >> a;

cin.ignore();

cout << "Enter name : ";

cin.getline(name, sizeof(name));

}

void show()

{

cout << a << " --- " << name << endl;

}

};

int main()

{

Test test;

// test.get();

// cout << "Enter name : ";

// cin.getline(name, sizeof(name));

//

// // writing file

// ofstream fout;

// fout.open("test.txt", ios::out);

//

// fout.write((char \*) &test, sizeof(test));

//

// fout.close();

//reading file

ifstream fin;

fin.open("test.txt", ios::in);

fin.read((char \*) &test, sizeof(test));

test.show();

fin.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll\_no;

char name[20];

public:

void get();

void show();

int get\_roll\_no()

{

return roll\_no;

}

};

void Student::get()

{

cout<<"enter roll no : ";

cin>>roll\_no;

cout<<"enter name : ";

cin>>name;

}

void Student::show()

{

cout<<"roll no : "<<roll\_no<<endl;

cout<<"name : "<<name<<endl;

}

int main()

{

// Student student[5];

//

// ofstream fout;

//

// fout.open("student.dat",ios::out);

//

// for(int i=0;i<5;i++)

// {

// student[i].get();

// fout.write((char\*) &student[i],sizeof(student[i]));

// }

// fout.close();

//1. find the second last object details

Student student;

ifstream file;

file.open("student.dat",ios::in);

//

// // file.seekg(48);

//

// file.seekg(-2\*sizeof(Student), ios::end);

// //

// file.read((char\*) &student, sizeof(student));

// // // file.read((char\*) &student, sizeof(student));

// // //

// int pos = file.tellg();

// // // //file.seekg(-pos, ios::cur);

// // // //file.read((char\*) &student, sizeof(student));

// student.show();

// // //

// cout << pos << endl;

//

while(file.read((char\*) &student, sizeof(student)))

{

if(student.get\_roll\_no() == 5)

{

cout << file.tellg() / sizeof(student)<< endl;

student.show();

}

}

//2. find the roll number of 3rd student

// file.seekg(0, ios::end);

// file.seekg(-1\*sizeof(student), ios::cur);

// student.show();

file.close();

//3. find position of roll number 77

return 0;

}

🡪#include <iostream>

#include <fstream>

#include <stdio.h>

using namespace std;

class Student

{

int roll\_no;

char name[20];

public:

void get();

void show();

int get\_roll\_no()

{

return roll\_no;

}

};

void Student::get()

{

cout<<"enter roll no : ";

cin>>roll\_no;

cout<<"enter name : ";

cin>>name;

}

void Student::show()

{

cout<<"roll no : "<<roll\_no<<endl;

cout<<"name : "<<name<<endl;

}

int main()

{

Student student;

Student tempstud;

//

// cout << "Enter data : " << endl;

// tempstud.get();

ifstream fin;

ofstream fout;

fin.open("temp.dat", ios::in);

// fout.open("temp.dat", ios::out);

while (fin.read((char \*) &student, sizeof(student)))

{

student.show();

}

// while (fin.read((char \*) &student, sizeof(student)))

// {

// if(tempstud.get\_roll\_no() > student.get\_roll\_no())

// {

// //write from existing file

// fout.write((char \*) &student, sizeof(student));

// }

// else if(tempstud.get\_roll\_no() < student.get\_roll\_no())

// {

// fout.write((char \*) &tempstud, sizeof(tempstud));

// fout.write((char \*) &student, sizeof(student));

// break;

// }

// }

//

// while (fin.read((char \*) &student, sizeof(student)))

// {

// fout.write((char \*) &student, sizeof(student));

// }

//

//

// remove("student.dat");

//

// rename("temp.dat", "student.dat");

fin.close();

fout.close();

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

int main()

{

int count = 0;

char line[80];

char word[10];

ifstream fin;

fin.open("test.txt", ios::in);

while(fin >> word)

{

cout << "word : " << word << endl;

count++;

}

cout << count << endl;

return 0;

}

🡪#include <iostream>

#include <fstream>

#include<string.h>

using namespace std;

int ismidg(char word[])

{

int mid;

mid = (strlen(word) + 1)/2;

if( strlen(word) % 2 != 0)

{

if(word[mid] == 'g')

return 1;

}

return 0;

}

int midg()

{

char word[20];

int count = 0;

//reading part

ifstream file;

file.open("file.txt", ios::in);

while(file >> word)

{

if(ismidg(word))

count++;

}

//remainder section of code

file.close();

return count;

}

int main()

{

cout << midg() << endl;

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll\_no;

char name[20];

public:

void get();

void show();

int get\_roll\_no()

{

return roll\_no;

}

};

void Student::get()

{

cout<<"enter roll no : ";

cin>>roll\_no;

cout<<"enter name : ";

cin>>name;

}

void Student::show()

{

cout<<"roll no : "<<roll\_no<<endl;

cout<<"name : "<<name<<endl;

}

int main()

{

return 0;

}

🡪//template for line to line or character to character type questions of file handling

#include <iostream>

#include <fstream>

using namespace std;

int notA()

{

char line[100];

int count = 0;

//reading part

ifstream file;

file.open("notA.txt", ios::in);

file.seekg(400,ios::beg);

while(file.getline(line, sizeof(line)))

{

if(line[0] != 'A')

count++;

}

//remainder section of code

file.close();

return count;

}

int main()

{

cout << notA() << endl;

return 0;

}

🡪#include <iostream>

#include <fstream>

using namespace std;

class Student

{

int roll\_no;

char name[20];

public:

void get();

void show();

int get\_roll\_no()

{

return roll\_no;

}

};

void Student::get()

{

cout<<"enter roll no : ";

cin>>roll\_no;

cout<<"enter name : ";

cin>>name;

}

void Student::show()

{

cout<<"roll no : "<<roll\_no<<endl;

cout<<"name : "<<name<<endl;

}

int main()

{

//search data of given roll number

int r;

cout << "Enter roll number : ";

cin >> r;

int flag = 0;

Student student;

ifstream fin;

fin.open("student.dat", ios::in);

while (fin.read((char \*) &student, sizeof(student)))

{

if(student.get\_roll\_no() == r)

{

flag = 1;

student.show();

}

}

if(!flag)

{

cout << "no such roll\_no." << endl;

}

fin.close();

return 0;

}

🡪#include<iostream>

#include<fstream>

#include <string.h>

using namespace std;

int isyou\_and\_me(char \* word)

{

if(strcmp(word,"you") == 0 || strcmp(word,"me") == 0)

return 1;

else

return 0;

}

int you\_and\_me()

{

char word[20];

int count=0;

ifstream file;

file.open("story.txt",ios::in);

while(file >> word)

{

if(isyou\_and\_me(word))

count++;

}

file.close();

return count;

}

int main()

{

cout<<you\_and\_me()<<endl;

}

🡪#include <iostream>

#include <fstream>

#include <ctype.h>

#include <string.h>

using namespace std;

int isTower(char \* word)

{

if(strcmp(word, "tower") == 0)

return 1;

else

return 0;

}

int tower()

{

char word[20];

int count = 0;

//reading part

ifstream file;

file.open("file.txt", ios::in);

while(file >> word)

{

if(strcmp(word, "tower") == 0)

count++;

}

//remainder section of code

file.close();

return count;

}

int main()

{

cout << tower() << endl;

return 0;

}

* Arrays

🡪#include <iostream>

using namespace std;

void multiply(int a[10][10], int b[10][10], int c[10][10], int m, int n, int p, int q, int &row, int &col)

{

if(n != p)

{

return;

}

row = m;

col = q;

int sum;

for(int x = 0; x < m; x++)

{

for(int y = 0; y < q; y++)

{

sum = 0;

for(int k = 0; k < p; k++)

{

sum += a[x][k] \* b[k][y];

}

c[x][y] = sum;

}

}

}

void print(int m[10][10], int rows, int cols)

{

for(int i = 0; i < rows; i++)

{

for(int j = 0; j < cols; j++)

{

cout << m[i][j] << "\t";

}

cout << endl;

}

}

void input(int m[10][10], int &rows, int &cols)

{

cout << "Enter rows : "; cin >> rows;

cout << "Enter columns : "; cin >> cols;

for(int i = 0; i < rows; i++)

{

for(int j = 0; j < cols; j++)

{

cin >> m[i][j];

}

}

}

int main()

{

int a[10][10], b[10][10], c[10][10];

int m, n, p, q, x, y;

cout << "Enter a : " << endl;

input(a, m, n);

cout << "Enter a : " << endl;

input(b, p, q);

multiply(a, b, c, m, n, p, q, x, y);

cout << "Result : " << endl;

print(c, x, y);

return 0;

}

🡪#include <iostream>

using namespace std;

void printArray(int \*a, int size)

{

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

{

cout << a[i] << ", ";

}

cout << "!!!" << endl;

}

void merge(int \*a, int a\_size, int \*b, int b\_size, int \*c, int c\_size)

{

int i; //array a index

int j; //array b index

int k; //array c index

i = j = k = 0; //chain initialisation

//compare elements from a and b

//assign required value

//change the index of the respective array

while(i < a\_size && j < b\_size)

{

if(a[i] <= b[j])

{

c[k] = a[i];

i++;

}

else

{

c[k] = b[j];

j++;

}

k++;

}

//when array b is finished

//insert all elements of a

while(j >= b\_size && i < a\_size)

{

c[k] = a[i];

i++;

k++;

}

//when array a is finished

//insert all elements of b

while(j < b\_size && i >= a\_size)

{

c[k] = b[j];

j++;

k++;

}

}

int main()

{

int a[] = {1, 3, 5, 9};

int b[] = {2, 4, 6, 7, 8, 9, 10};

int c[11];

int a\_size = 4;

int b\_size = 7;

int c\_size = a\_size + b\_size;

merge(a, a\_size, b, b\_size, c, c\_size);

printArray(a, a\_size);

printArray(b, b\_size);

printArray(c, c\_size);

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void printArray(int a[], int start, int end)

{

cout << "-----------------------------------------------" << endl;

for(int i = start; i <= end; i++)

cout << a[i] << " ";

cout << endl;

cout << "-----------------------------------------------" << endl;

}

int bsearch(int a[], int size, int e)

{

int l = 0;

int h = size - 1;

int mid;

cout << "size = " << size << endl;

while(l <= h)

{

printArray(a, l, h);

mid = (l + h) / 2;

cout << endl << mid << " - > " << a[mid] << endl;

if(a[mid] == e)

return mid;

else if(a[mid] < e)

l = mid + 1;

else

h = mid - 1;

}

return -1;

}

int main()

{

int a[] = {0, 1, 2, 3, 4, 7, 89, 456, 553, 2354, 10023};

int size = sizeof(a) / sizeof(a[0]);

cout << endl << bsearch(a, size, 456) << endl;

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void printArray(int a[], int start, int end)

{

cout << "-----------------------------------------------" << endl;

for(int i = start; i < end; i++)

cout << a[i] << " ";

cout << endl;

cout << "-----------------------------------------------" << endl;

}

void swap(int &a, int &b)

{

int temp;

temp = a;

a = b;

b = temp;

}

void ascBubbleSort(int a[], int size)

{

for(int pass = 1; pass < size; pass++)

{

printArray(a, 0, size);

for(int j = 0; j < size - pass; j++)

{

if(a[j] > a[j+1])

{

swap(a[j], a[j+1]);

}

}

}

}

void descBubbleSort(int a[], int size)

{

for(int pass = 1; pass < size; pass++)

{

printArray(a, 0, size);

for(int j = 0; j < size - pass; j++)

{

if(a[j] < a[j+1])

{

swap(a[j], a[j+1]);

}

}

}

}

int main()

{

int a[] = {8, 1, 3, 0, 6, 9, 2};

int size = sizeof(a) / sizeof(a[0]);

descBubbleSort(a, size);

return 0;

}

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void printArray(int a[], int start, int end)

{

cout << "-----------------------------------------------" << endl;

for(int i = start; i < end; i++)

cout << a[i] << " ";

cout << endl;

cout << "-----------------------------------------------" << endl;

}

void swap(int &a, int &b)

{

int temp;

temp = a;

a = b;

b = temp;

}

void ascInsertionSort(int a[], int size)

{

for(int i = 1; i <= size; i++)

{

printArray(a, 0, size);

for(int j = 0; j <= i; j++)

{

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

{

swap(a[j], a[i]);

}

}

}

}

void descInsertionSort(int a[], int size)

{

for(int i = 1; i <= size; i++)

{

printArray(a, 0, size);

for(int j = 0; j <= i; j++)

{

if(a[j] < a[i])

{

swap(a[j], a[i]);

}

}

}

}

int main()

{

int a[] = {INT\_MIN, 9, 8, 1, 0, 9, 1, 0, 4, 9, 2, 8, 1, 3, 0, 6, 9, 2, 8, 4, 3};

int size = sizeof(a) / sizeof(a[0]);

ascInsertionSort(a, size);

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

void printArray(int a[], int start, int end)

{

cout << "-----------------------------------------------" << endl;

for(int i = start; i < end; i++)

cout << a[i] << " ";

cout << endl;

cout << "-----------------------------------------------" << endl;

}

void swap(int &a, int &b)

{

int temp;

temp = a;

a = b;

b = temp;

}

void ascendingSelectionSort(int a[], int size)

{

int min = 0; //index of the minimum element

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

{

printArray(a, 0, size);

min = i;

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

{

if(a[min] >= a[j])

{

min = j;

}

}

swap(a[min], a[i]);

}

printArray(a, 0, size);

}

void descendingSelectionSort(int a[], int size)

{

int max = 0; //index of the minimum element

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

{

printArray(a, 0, size);

max = i;

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

{

if(a[max] < a[j])

{

max = j;

}

}

swap(a[max], a[i]);

}

printArray(a, 0, size);

}

int main()

{

int a[] = {9, 8, 1, 0, 9, 1, 0, 4, 9, 2, 8, 1, 3, 0, 6, 9, 2, 8, 4, 3};

int size = sizeof(a) / sizeof(a[0]);

descendingSelectionSort(a, size);

return 0;

}

🡪

* Linked lists , stack and queues

🡪#include <iostream>

using namespace std;

typedef struct Node \* Nodeptr;

struct Node

{

int data;

Nodeptr next;

};

//global variable

Nodeptr start = NULL;

Nodeptr createNode(int x)

{

Nodeptr temp = new Node;

temp->data = x;

temp->next = NULL;

return temp;

}

void insert\_beginning(int x)

{

Nodeptr temp;

temp = createNode(x);

if(start == NULL)

{

start = temp;

return;

}

temp->next = start;

start = temp;

return;

}

void insert\_end(int x)

{

Nodeptr temp;

temp = createNode(x);

//if list is empty

if(start == NULL)

{

start = temp;

return;

}

Nodeptr p;

p = start;

while(p->next != NULL)

{

p = p->next;

}

p->next = temp;

}

int deletion\_beginning()

{

//if list is empty

if(start == NULL)

{

cout << "List is empty." << endl;

return -1;

}

Nodeptr temp;

int number;

temp = start;

start = start->next;

number = temp->data;

delete temp;

return number;

}

void printList()

{

Nodeptr p; //traversal pointer

p = start;

while(p != NULL)

{

cout << p->data << " -> ";

p = p->next;

}

cout << "!!!" << endl;

}

int main()

{

int x = 0;

cout << "Enter elements : " << endl;

while(true)

{

cin >> x;

if(x == -1)

break;

insert\_end(x);

}

printList();

cout << "deleted data = " << deletion\_beginning() << endl;

printList();

return 0;

}

🡪#include <iostream>

using namespace std;

typedef struct Node \* nodePointer;

struct Node

{

int data;

nodePointer link;

};

//global variable

nodePointer start = NULL;

nodePointer createNode(int x)

{

nodePointer temp;

temp = new struct Node;

temp->data = x;

temp->link = NULL;

return temp;

}

void insert\_beginning(int x)

{

nodePointer temp;

temp = createNode(x);

if(start == NULL)

{

//list is empty

start = temp;

return;

}

temp->link = start;

start = temp;

}

void insert\_end(int x)

{

nodePointer temp;

temp = createNode(x);

if(start == NULL)

{

//list is empty

start = temp;

return;

}

nodePointer p; //traversal pointer

p = start;

while(p->link != NULL)

{

p = p->link;

}

p->link = temp;

}

int deletion\_beginning()

{

if(start == NULL)

{

return -1;

}

nodePointer temp;

temp = start;

start = start->link;

int x = temp->data;

delete temp;

return x;

}

int deletion\_end()

{

if(start == NULL)

{

cout << "list is empty." << endl;

return -1;

}

nodePointer p; //traversal nodePointer

p = start;

int x;

if(p->link == NULL)

{

//only one node

start = NULL;

x = p->data;

delete p;

return x;

}

while(p->link->link != NULL && p->link != NULL)

{

p = p->link;

}

nodePointer temp;

temp = p->link;

x = temp->data;

delete temp;

p->link = NULL;

return x;

}

void printList()

{

nodePointer p; //traversal pointer

p = start;

while(p != NULL)

{

cout << p->data << " -> ";

p = p->link;

}

cout << "!!!" << endl;

}

int main()

{

int x = 0;

while(x != -1)

{

cin >> x;

if(x == -1)

break;

insert\_end(x);

}

printList();

cout << "deleted element = " << deletion\_end() << endl;

printList();

return 0;

}

🡪e#include <iostream>

using namespace std;

struct Stack

{

int data;

struct Stack \*next;

};

//top pointer

struct Stack \*top = NULL;

void push(int x)

{

Stack \*temp;

temp = new struct Stack;

temp->data = x;

temp->next = NULL;

if(top == NULL)

{

top = temp;

return;

}

temp->next = top;

top = temp;

}

int pop()

{

int x = -1;

if(top == NULL)

{

return x;

}

Stack \*temp;

temp = top;

x = temp->data;

top = top->next;

delete temp;

return x;

}

int peak()

{

int x = -1;

if(top == NULL)

{

return x;

}

x = top->data;

return x;

}

void printStack()

{

Stack \*p;

p = top;

while(p != NULL)

{

cout << p->data << endl;

p = p->next;

}

cout << "\_\_\_\_\_\_\_\_" << endl;

}

int main()

{

int number;

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. push" << endl;

cout << "2. pop" << endl;

cout << "3. peak" << endl;

cout << "4. print stack" << endl;

cout << "5. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter number : "; cin >> number;

push(number);

cout << number << " pushed in the list." << endl;

break;

case 2:

number = pop();

if(number != -1)

cout << number << " popped." << endl;

else

cout << "Stack is empty." << endl;

break;

case 3:

if(peak() != -1)

cout << peak() << " is the peak element." << endl;

else

cout << "Stack is empty." << endl;

break;

case 4:

printStack();

break;

}

}

while(choice != 5);

cout << "program exited." << endl;

return 0;

}

🡪//implementation of stack with complex data using class

#include <iostream>

#include <string.h>

using namespace std;

class Student

{

int roll\_number;

char name[20];

public:

Student()

{

roll\_number = 0;

strcpy(name, "NONE");

}

Student(Student &obj)

{

roll\_number = obj.roll\_number;

strcpy(name, obj.name);

cout << "Copy constructor called." << endl;

}

void get()

{

cout << "Enter roll number : ";

cin >> roll\_number;

cin.ignore();

cout << "Enter name : ";

cin.getline(name, sizeof(name));

}

void show()

{

cout << "--------------------------------" << endl;

cout << "Roll number : " << roll\_number << endl;

cout << "Name : " << name << endl;

cout << "--------------------------------" << endl;

}

};

struct Node

{

Student data;

struct Node \*next;

};

class Stack

{

struct Node \*top;

public:

Stack()

{

top = NULL;

}

void push(Student x);

void pop();

Student peak();

void printStack();

~Stack()

{

cout << "Stack pointer deleted." << endl;

Node \*temp;

while(top != NULL)

{

temp = top;

top = top->next;

delete temp;

}

delete top;

}

};

//member function definitions

void Stack::push(Student x)

{

Node \*temp;

temp = new struct Node;

temp->data = x;

temp->next = NULL;

if(top == NULL)

{

top = temp;

return;

}

temp->next = top;

top = temp;

}

void Stack::pop()

{

Student x;

if(top == NULL)

{

cout << "No Student in the Stack!" << endl;

return;

}

//cout << "here" << endl;

Node \*temp;

temp = top;

x = temp->data; // due to copy constructor

top = top->next;

delete temp;

cout << "Data of Student Deleted : " << endl;

x.show();

}

Student Stack::peak()

{

Student x;

if(top == NULL)

{

return x;

}

x = top->data;

return x;

}

void Stack::printStack()

{

Node \*p;

p = top;

while(p != NULL)

{

(p->data).show();

p = p->next;

}

cout << "\_\_\_\_\_\_\_\_" << endl;

}

int main()

{

Stack s;

Student object;

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. push" << endl;

cout << "2. pop" << endl;

cout << "3. peak" << endl;

cout << "4. print stack" << endl;

cout << "5. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter Data of student : " << endl;

object.get();

s.push(object);

break;

case 2:

s.pop();

//cout << "here" << endl;

break;

case 3:

object = s.peak();

cout << "Data on the top : " << endl;

object.show();

break;

case 4:

s.printStack();

break;

}

}

while(choice != 5);

cout << "program exited." << endl;

return 0;

}

🡪#include <iostream>

#include <string.h>

using namespace std;

struct Node

{

char data[20];

Node \*next;

};

class Stack

{

Node \*top;

public:

Stack()

{

top = NULL;

cout << "Constructor called." << endl;

}

void push(char \*x)

{

//Node creation

Node \*temp;

temp = new Node;

strcpy(temp->data, x);

temp->next = NULL;

//check if stack is empty or not

if(top == NULL)

{

cout << "Stack was empty." << endl;

top = temp;

return;

}

//stack is not empty

temp->next = top;

top = temp;

}

char \*pop()

{

if(top == NULL)

{

cout << "Stack is empty." << endl;

return "NONE";

}

char \*x;

x = new char[20];

Node \*temp;

temp = top;

strcpy(x, temp->data);

top = top->next;

delete temp;

return x;

}

char \*peak()

{

if(top == NULL)

{

cout << "Stack is empty." << endl;

return "NONE";

}

Node \*temp;

temp = top;

return temp->data;

}

void printStack()

{

Node \*p;

p = top;

while(p != NULL)

{

cout << p->data << endl;

p = p->next;

}

cout << "\_\_\_\_\_\_\_\_\_" << endl;

}

};

int main()

{

Stack s;

char x[20];

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. push" << endl;

cout << "2. pop" << endl;

cout << "3. peak" << endl;

cout << "4. print stack" << endl;

cout << "5. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter string : " << endl;

cin.ignore();

cin.getline(x, sizeof(x));

s.push(x);

cout << x << " is pushed to the stack." << endl;

break;

case 2:

strcpy(x, s.pop());

cout << x << " is popped from the stack." << endl;

break;

case 3:

strcpy(x, s.peak());

cout << "Data on the top : " << x << endl;

break;

case 4:

s.printStack();

break;

}

}

while(choice != 5);

cout << "program exited." << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

struct Employee

{

int eno;

char name[20];

};

struct Node

{

Employee data;

Node \* next;

};

class Stack

{

Node \* top;

public:

Stack()

{

top = NULL;

}

void push(Employee x);

void pop();

void peak();

void printStack();

};

void Stack::push(Employee x)

{

//creation of the Node

Node \* temp;

temp = new Node;

temp->data = x;

temp->next = NULL;

//if Stack is empty

if(top == NULL)

{

top = temp;

return;

}

//if stack is not empty

temp->next = top;

top = temp;

}

void Stack::pop()

{

if(top == NULL)

{

cout << "Underflow!" << endl;

return;

}

Node \* temp;

temp = top;

top = top->next;

cout << "Data deleted is : " << endl;

cout << "Eno = " << (temp->data).eno << endl;

cout << "Name = " << (temp->data).name << endl;

delete temp;

}

void Stack::peak()

{

if(top == NULL)

{

cout << "Underflow!" << endl;

return;

}

cout << "Data on the top is : " << endl;

cout << "Eno = " << (top->data).eno << endl;

cout << "Name = " << (top->data).name << endl;

}

void Stack::printStack()

{

Node \* p;

for(p = top; p != NULL; p = p->next)

{

cout << "-------------------------------" << endl;

cout << "Eno = " << (p->data).eno << endl;

cout << "Name = " << (p->data).name << endl;

cout << "-------------------------------" << endl;

}

cout << "+-------------------------------------------+" << endl;

}

int main()

{

Stack s;

Employee x;

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. push" << endl;

cout << "2. pop" << endl;

cout << "3. peak" << endl;

cout << "4. print stack" << endl;

cout << "5. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter Data : " << endl;

cout << "Enter eno : "; cin >> x.eno;

cin.ignore();

cout << "Enter name : "; cin.getline(x.name, sizeof(x.name));

s.push(x);

cout << "Data is pushed to the stack." << endl;

break;

case 2:

s.pop();

break;

case 3:

s.peak();

break;

case 4:

s.printStack();

break;

}

}

while(choice != 5);

cout << "program exited." << endl;

return 0;

}

🡪#include <iostream>

#define MAX 10

using namespace std;

class Stack

{

int top;

int data[MAX];

public:

Stack()

{

top = -1;

}

void push(int x)

{

if(top == MAX - 1)

{

cout << "Overflow!" << endl;

return;

}

top++;

data[top] = x;

}

void pop()

{

if(top == -1)

{

cout << "Underflow!" << endl;

return;

}

cout << data[top] << " popped." << endl;

top--;

}

void peak()

{

if(top == -1)

{

cout << "Underflow!" << endl;

return;

}

cout << data[top] << " on the top." << endl;

}

void printStack()

{

for(int i = top; i >= 0; i--)

{

cout << data[i] << endl;

}

cout << "\_\_\_\_\_\_\_" << endl;

}

};

int main()

{

Stack s;

int x;

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. push" << endl;

cout << "2. pop" << endl;

cout << "3. peak" << endl;

cout << "4. print stack" << endl;

cout << "5. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter Data : " << endl;

cin >> x;

s.push(x);

break;

case 2:

s.pop();

//cout << "here" << endl;

break;

case 3:

s.peak();

break;

case 4:

s.printStack();

break;

}

}

while(choice != 5);

cout << "program exited." << endl;

}

🡪#include <iostream>

#include <string.h>

using namespace std;

//node definition

struct Node

{

char \*data;

Node \*next;

};

class Queue

{

Node \*front;

Node \*rear;

public:

Queue()

{

front = NULL;

rear = NULL;

}

void enqueue(char \*x);

char \*dequeue();

void printQueue();

};

void Queue::enqueue(char \*x)

{

//node creation

Node \*temp;

temp = new Node;

temp->data = new char[strlen(x) + 1];

strcpy(temp->data, x);

temp->next = NULL;

//if queue is empty

if(front == NULL && rear == NULL)

{

front = temp;

rear = temp;

return;

}

//if queue is not empty

rear->next = temp;

rear = temp;

}

char \* Queue::dequeue()

{

if(front == NULL && rear == NULL)

{

cout << "UnderFlow !" << endl;

return "NONE";

}

char \*x;

x = front->data;

Node \*temp;

temp = front;

front = front->next;

delete temp;

return x;

}

void Queue::printQueue()

{

Node \* p; //traversal pointer

p = front;

while(p != NULL)

{

cout << p->data << " <-- ";

p = p->next;

}

cout << "!!!" << endl;

}

int main()

{

Queue q;

char word[50];

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. enqueue" << endl;

cout << "2. dequeue" << endl;

cout << "3. print queue" << endl;

cout << "4. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter Word to the queue : " << endl;

cin.ignore();

cin.getline(word, sizeof(word));

q.enqueue(word);

cout << "new word added to the queue" << endl;

break;

case 2:

strcpy(word, q.dequeue());

cout << word << " deleted from the queue." << endl;

break;

case 3:

q.printQueue();

break;

}

}

while(choice != 4);

cout << "program exited." << endl;

return 0;

}

🡪qu#include <iostream>

#include <string.h>

using namespace std;

class Student

{

int roll\_number;

char name[20];

public:

Student()

{

roll\_number = 0;

strcpy(name, "NONE");

}

Student(Student &obj)

{

roll\_number = obj.roll\_number;

strcpy(name, obj.name);

cout << "Copy constructor called." << endl;

}

void get()

{

cout << "Enter roll number : ";

cin >> roll\_number;

cin.ignore();

cout << "Enter name : ";

cin.getline(name, sizeof(name));

}

void show()

{

cout << "--------------------------------" << endl;

cout << "Roll number : " << roll\_number << endl;

cout << "Name : " << name << endl;

cout << "--------------------------------" << endl;

}

~Student()

{

cout << "Destructor called." << endl;

}

};

//node definition

struct Node

{

Student data;

Node \*next;

};

class Queue

{

Node \*front;

Node \*rear;

public:

Queue()

{

front = NULL;

rear = NULL;

}

void enqueue(Student x);

void dequeue();

void printQueue();

};

void Queue::enqueue(Student x)

{

//node creation

Node \*temp;

temp = new Node;

temp->data = x;

temp->next = NULL;

//if queue is empty

if(front == NULL && rear == NULL)

{

front = temp;

rear = temp;

return;

}

//if queue is not empty

rear->next = temp;

rear = temp;

}

void Queue::dequeue()

{

if(front == NULL && rear == NULL)

{

cout << "UnderFlow !" << endl;

return;

}

Node \*temp;

temp = front;

if(front == rear)

{

rear = rear->next;

}

front = front->next;

cout << "Data to be deleted is : " << endl;

(temp->data).show();

delete temp;

}

void Queue::printQueue()

{

Node \* p; //traversal pointer

p = front;

while(p != NULL)

{

(p->data).show();

p = p->next;

}

cout << "!!!!!!!!!!!!!!!" << endl;

}

int main()

{

Queue q;

Student x;

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. enqueue" << endl;

cout << "2. dequeue" << endl;

cout << "3. print queue" << endl;

cout << "4. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter Data to the queue : " << endl;

x.get();

q.enqueue(x);

cout << "new word added to the queue" << endl;

break;

case 2:

q.dequeue();

break;

case 3:

q.printQueue();

break;

}

}

while(choice != 4);

cout << "program exited." << endl;

return 0;

}

🡪#include <iostream>

using namespace std;

//data is a structure

struct Employee

{

int eno;

char name[20];

float salary;

};

struct Node

{

Employee data;

Node \*next;

};

class Queue

{

Node \*front;

Node \*rear;

public:

Queue()

{

front = NULL;

rear = NULL;

}

void enqueue(Employee x)

{

//insert to the end

//node creation

Node \*temp;

temp = new Node;

temp->data = x;

temp->next = NULL;

//if queue is empty?

if(front == NULL && rear == NULL)

{

front = temp;

rear = temp;

return;

}

rear->next = temp;

rear = temp;

}

void dequeue()

{

if(front == NULL && rear == NULL)

{

cout << "Underflow!" << endl;

return;

}

Node \*temp;

temp = front;

if(front == rear)

{

front = NULL;

rear = NULL;

cout << "Data deleted is : " << endl;

cout << "Employee number = " << (temp->data).eno << endl;

cout << "Employee name = " << (temp->data).name << endl;

cout << "Employee salary = " << (temp->data).salary << endl;

delete temp;

return;

}

front = front->next;

cout << "Data deleted is : " << endl;

cout << "Employee number = " << (temp->data).eno << endl;

cout << "Employee name = " << (temp->data).name << endl;

cout << "Employee salary = " << (temp->data).salary << endl;

delete temp;

}

void printQueue()

{

Node \*p;

p = front;

while(p != NULL)

{

cout << "------------------------------------------------" << endl;

cout << "Employee number = " << (p->data).eno << endl;

cout << "Employee name = " << (p->data).name << endl;

cout << "Employee salary = " << (p->data).salary << endl;

cout << "------------------------------------------------" << endl;

p = p->next;

}

}

};

int main()

{

Queue q;

Employee x;

int choice;

do

{

/\* code \*/

cout << "-==========================================-" << endl;

cout << "MENU" << endl;

cout << "1. enqueue" << endl;

cout << "2. dequeue" << endl;

cout << "3. print queue" << endl;

cout << "4. exit" << endl;

cout << "Enter choice : " << endl;

cin >> choice;

switch(choice)

{

case 1:

cout << "Enter Data to the queue : " << endl;

cout << "Enter eno : "; cin >> x.eno;

cin.ignore();

cout << "Enter name : "; cin.getline(x.name, sizeof(x.name));

cout << "Enter salary : "; cin >> x.salary;

q.enqueue(x);

cout << "new data added to the queue" << endl;

break;

case 2:

q.dequeue();

break;

case 3:

q.printQueue();

break;

}

}

while(choice != 4);

cout << "program exited." << endl;

return 0;

}

🡪#include <iostream>

#include <bits/stdc++.h>

using namespace std;

struct Node

{

char data;

Node \*next;

};

class Stack

{

Node \*top;

public:

Stack()

{

top = NULL;

}

void push(char x)

{

Node \*temp;

temp = new Node;

temp->next = NULL;

temp->data = x;

if(top == NULL)

{

top = temp;

return;

}

temp->next = top;

top = temp;

}

char pop()

{

char x = top->data;

Node \*temp;

temp = top;

top = top->next;

delete temp;

return x;

}

int isEmpty()

{

if(top == NULL)

return 1;

return 0;

}

char peak()

{

char x = top->data;

return x;

}

};

int precedence(char symbol)

{

switch (symbol)

{

case '+': return 1;

case '-': return 1;

case '\*': return 2;

case '/': return 2;

case '^': return 3;

}

}

void changeToPostFix(char \* input)

{

Stack stack;

char output[50];

int k = 0;

for(int i = 0; input[i] != NULL; i++)

{

char scannedSymbol = input[i];

if(isalpha(scannedSymbol))

{

output[k++] = scannedSymbol;

continue;

}

else if(stack.isEmpty() == 1)

{

stack.push(scannedSymbol);

continue;

}

else if(scannedSymbol == '(')

{

stack.push(scannedSymbol);

continue;

}

else if(scannedSymbol == ')')

{

while(stack.peak() != '(')

{

char poppedSymbol = stack.pop();

output[k++] = poppedSymbol;

}

stack.pop();

}

else if(precedence(scannedSymbol) > precedence(stack.peak()))

{

stack.push(scannedSymbol);

continue;

}

else if(precedence(scannedSymbol) <= precedence(stack.peak()))

{

while(stack.isEmpty() != 1 && precedence(scannedSymbol) <= precedence(stack.peak()))

{

char poppedSymbol = stack.pop();

output[k++] = poppedSymbol;

}

stack.push(scannedSymbol);

continue;

}

}

while(stack.isEmpty() != 1)

{

char poppedSymbol = stack.pop();

output[k++] = poppedSymbol;

}

cout << output << endl;

}

int main()

{

char input[50];

cout << "Enter infix string : " << endl;

cin >> input;

cout << "Postfix string : " << endl;

changeToPostFix(input);

return 0;

}