**RESULT:**

**RESULT:**

**RESULT:**

**second\_no=next\_no;**

**}**

**return 0;**

**}**

**RESULT:**

**RESULT:**

**EXPERIMENT NO:2.a**

**Swap with variable**

**AIM:**

### **Algorithm**

1. **Start**
2. **Initialize** two integers n1 and n2 with values (25 and 50).
3. **Print** the values of n1 and n2 before the swap.
4. **Call** the swap function, passing the addresses of n1 and n2.
5. In the swap function:
   1. Create a temporary variable temp.
   2. Store the value pointed by n1 in temp.
   3. Assign the value pointed by n2 to the location pointed by n1.
   4. Assign temp to the location pointed by n2.
6. **Return** to the main function.
7. **Print** the values of n1 and n2 after the swap.
8. **End**

**SOURCE CODE**

**#include<iostream>**

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

**using namespace std;**

**int main()**

**{**

**int n1=25,n2=50;**

**cout<<"Before swap:"<<endl<<"Number1="<<n1<<endl<<"Number2="<<n2<<endl;**

**swap(&n1,&n2);**

**cout<<"After swap:"<<endl<<"Number1="<<n1<<endl<<"Number2="<<n2<<endl;**

**return 0;**

**}**

**void swap(int\*n1,int\*n2)**

**{**

**int temp;**

**temp=\*n1;**

**OUTPUT:**

**Before swap:**

**Number1 = 25**

**Number2 = 50**

**After swap:**

**Number1 = 50**

**Number2 = 25**

**\*n2=temp;**

**}**

**RESULT:**

**EXPERIMENT NO : 2.b**  **DATE :**

**SWAP WITHOUT VARIABLE**

**AIM:**

### **Algorithm**

1. **Start**
2. **Initialize** two integers, n1 and n2, with values 25 and 50.
3. **Print** the values of n1 and n2 before the swap.
4. **Call** the swap function, passing the addresses of n1 and n2.
5. Inside the swap function:
   1. Compute the sum of \*n1 and \*n2 and store it in \*n1.
   2. Update \*n2 to the original value of \*n1 by subtracting the new \*n2.
   3. Update \*n1 to the original value of \*n2 by subtracting the updated \*n2.
6. **Return** to the main function.
7. **Print** the values of n1 and n2 after the swap.
8. **End**

**SOURCE CODE:**

**#include<iostream>**

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

**using namespace std;**

**int main()**

**{**

**int n1=25,n2=50;**

**cout<<"Before swap:"<<endl<<"Number1="<<n1<<endl<<"Number2="<<n2<<endl;**

**swap(&n1,&n2);**

**cout<<"After swap:"<<endl<<"Number1="<<n1<<endl<<"Number2="<<n2<<endl;**

**return 0;**

**}**

**void swap(int\*n1,int\*n2)**

**{**

**\*n1=\*n1+\*n2;**

**\*n2=\*n2-\*n2;**

**\*n2=\*n1-\*n2;**

**OUTPUT:**

**Before swap:**

**Number1 = 25**

**Number2 = 50**

**After swap:**

**Number1 = 50**

**Number2 = 25**

**}**

**Result:**

**EXPERIMENT NO : 2.c** **DATE :**

**CUBE**

**AIM:**

### **Algorithm**

1. **Start**
2. **Declare Variables**:
   1. int number: to store the user-input number.
   2. int \*ptr: a pointer to int, used to point to number.
   3. int cube: to store the calculated cube of the number.
3. **Assign Pointer**:
   1. Set ptr to the address of number using ptr = &number;.
4. **Input**:
   1. Prompt the user to enter a number with the message "Enter a number:".
   2. Read the value from the user and store it in the location pointed to by ptr using cin >> \*ptr;.
5. **Calculate Cube**:
   1. Compute the cube of the number using the expression cube = (\*ptr) \* (\*ptr) \* (\*ptr);.
6. **Output**:
   1. Display the result of the cube with the message "cube = " followed by the calculated cube.
7. **End**

**SOURCE CODE**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int number,\*ptr,cube;**

**ptr=&number;**

**cout<<"Enter a number:";**

**cin>>\*ptr;**

**cube=(\*ptr)\*(\*ptr)\*(\*ptr);**

**cout<<"cube="<<cube;**

**return 0;**

**OUTPUT:**

**Enter a number: 3**

**cube = 27**

**}**

**RESULT:**

**EXPERIMENT NO: DATE:**

**SUM OF ARRAY**

**ALGOITHM**

1. **START**
2. **Initialization:**
   1. Declare an array a of size 100, an integer n for the size of the array, an integer pointer ptr pointing to the array, and variables i and sum initialized to 0.
3. **Input the Array Size:**
   1. Prompt the user to enter the size of the array and store it in n.
4. **Input Array Elements:**
   1. Prompt the user to enter n elements and store them in the array a.
5. **Calculate the Sum:**
   1. Use a loop to iterate over the elements of the array using the pointer ptr, and accumulate the sum of the elements into the variable sum.
6. **Output the Result:**
   1. Print the calculated sum.
7. **STOP**

**SOURCE CODE:**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**int a[100],n,\*ptr,i,sum=0;**

**ptr=a;**

**cout<<"Enter array size:";**

**cin>>n;**

**cout<<"Enter array elements:";**

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

**{**

**cin>>a[i];**

**}**

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

**{**

**sum=sum+(\*(ptr+i));**

**INPUT:**

**Enter array size: 5**

**Enter array elements: 1 2 3 4 5**

**OUTPUT:**

**Sum of Array = 15**

**}**

**cout<<"Sum of Array="<<sum;**

**return 0;**

**}**

**RESULT:**

**EXPERIMENT NO : 3.a**  **DATE :**

**INLINE**

**AIM:**

### **Algorithm**

**1. Start**

**2. Define class operation**

2.1. **Declare private members**: int n1, n2

2.2. **Declare public methods:**

- void get()

- void sum()

- void difference()

- void product()

- void division()

**3. Implement method get()**

**4. Implement method sum()**

**5. Implement method difference()**

**6. Implement method product()**

**7. Implement method division()**

**8. Define main function**

**9. Stop**

**SOURCE CODE:**

#include<iostream>

using namespace std;

class operation{

private:

int n1,n2;

public:

void get();

void sum();

void difference();

**OUTPUT:**

**Enter n1,n2: 10 5**

**Sum = 15**

**Difference = 5**

**Product = 50**

**Division = 2**

void product();

void division();

};

inline void operation::get(){

cout<<"Enter n1,n2:";

cin>>n1;

cin>>n2;

}

inline void operation::sum(){

cout<<"Sum ="<<n1+n2<<endl;

}

inline void operation::difference(){

cout<<"Difference ="<<n1-n2<<endl;

}

inline void operation::product(){

cout<<"Product ="<<n1\*n2<<endl;

}

inline void operation::division(){

cout<<"Division ="<<n1/n2<<endl;

}

int main()

{

operation op;

op.get();

op.sum();

op.difference();

op.product();

op.division();

return 0;

}

**EXPERIMENT NO : 3.b** **DATE :**

**FRIEND FUNCTION**

**AIM:**

### **Algorithm**

1. **Start**
2. **Define a Class: Area**
   1. **Private Variables:**
      1. int l (length of the rectangle)
      2. int b (breadth of the rectangle)
      3. int base (base of the triangle)
      4. int h (height of the triangle)
   2. **Public Methods:**
      1. void get\_r() (to get rectangle dimensions)
      2. void get\_t() (to get triangle dimensions)
      3. friend int area\_rectangle(int l, int b) (to calculate area of rectangle)
      4. friend float area\_triangle(int base, int h) (to calculate area of triangle)
3. **Implement Method: get\_r()**
4. **Implement Method: get\_t()**
5. **Implement Friend Function: area\_rectangle(int l, int b)**
6. **Implement Friend Function: area\_triangle(int base, int h)**
7. **Define Main Function**
8. **Stop**

**SOURCE CODE:**

**#include<iostream>**

**using namespace std;**

**class Area{**

**private:**

**int l,b;**

**int base,h;**

**public:**

**void get\_r();**

**friend int area\_rectangle(int l,int b);**

**void get\_t();**

**friend float area\_triangle(int base,int h);**

**}**

**OUTPUT:**

**Enter length and breadth: 5 10**

**Enter base and height: 6 8**

**Area of rectangle = 50**

**Area of triangle = 24**

**inline void Area::get\_r(){**

**cout<<"Enter length and breadth:";**

**cin>>l>>b;**

**}**

**inline void Area:get\_t(){**

**cout<<"Enter base and height:";**

**cin>>base>>h;**

**}**

**inline int area\_rectangle(int l,int b)**

**{**

**cout<<"Area of rectangle="<<l\*b;**

**}**

**inline int area\_triangle(int base,int h)**

**{**

**cout<<"Area of triangle="<<0.5\*base\*h;**

**}**

**int main()**

**{**

**Area a;**

**a.get\_r();**

**a.get\_t();**

**a.area\_rectangle();**

**a.area\_triangle();**

**return 0;**

**}**

**EXPERIMENT NO 3.c**   **DATE :**

**AIM:**

### **Algorithm**

1. **Start**
2. **Function Definition: prod(int n)**
   1. Declare a static variable prod initialized to 1.
   2. Multiply prod by n.
   3. Return the value of prod.
3. **Main Function Execution**
   1. Call prod(2) and output the result.
   2. Call prod(3) and output the result.
   3. Call prod(8) and output the result.
4. **Stop**

**STATIC**

**#include<iostream>**

**using namespace std;**

**int prod(int n)**

**{**

**static int prod=1;**

**prod\*=n;**

**return prod;**

**}**

**int main()**

**{**

**cout<<prod(2)<<endl;**

**cout<<prod(3)<<endl;**

**cout<<prod(8)<<endl;**

**}**

**OUTPUT:**

**2**

**6**

**48**

**Result:**

**EXPERIMENT NO : 4.a** **DATE :**

**CLASS AND OBJECTS**

**(STUDENT INFORMATION SYSTEM)**

**AIM:**

**ALGORITHM**

* **Start**
* **Define Class: Student**
* **Private Variables:**
  + int reg\_no (Registration number)
  + char name[20] (Name of the student)
  + int marks[5] (Marks in 5 subjects)
* **Public Method:**
  + void get() (to get student details)
* **Implement Method: get()**
* Declare variables int n, int sum, float avg.
* **Define Main Function**
* **Stop**

**SOURCE CODE:**

**#include<iostream>**

**using namespace std;**

**class Student**

**{**

**private:**

**int reg\_no,i,n;**

**char name[20];**

**int marks[5];**

**float a;**

**public:**

**void get();**

**}s;**

**void Student::get(){**

**int sum=0,avg;**

**cout<<"STUDENT INFORMATION SYSTEM:"<<endl;**

**OUTPUT:**

**STUDENT INFORMATION SYSTEM:**

**Enter number of students: 2**

**Enter Register number: 101**

**Enter Name: Alice**

**Enter 5 subject marks: 85 90 78 88 92**

**Student 1 Details:**

**Register Number: 101**

**Name: Alice**

**Marks: 85 90 78 88 92**

**Average: 86.6**

**Enter Register number: 102**

**Enter Name: Bob**

**Enter 5 subject marks: 75 80 70 65 60**

**Student 2 Details:**

**Register Number: 102**

**Name: Bob**

**Marks: 75 80 70 65 60**

**Average: 64**

**cout<<"Enter no of students:"<<endl;**

**cin>>n;**

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

**{**

**cout<<"Enter Register number:";**

**cin>>s.reg\_no;**

**cout<<"Enter Name:";**

**cin>>s.name;**

**}**

**cout<<"Enter 5 subject marks:"<<endl;**

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

**{**

**cin>>marks[i];**

**}**

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

**{**

**sum=sum+marks[i];**

**}**

**avg=sum/5;**

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

**cout<<"\nStudent"<<i+1;**

**cout<<"\nRegister name:"<<s.reg\_no;**

**cout<<"\nName:"<<s.name;**

**}**

**cout<<"Marks:"<<s.marks;**

**cout<<"Average:"<<avg;**

**}**

**int main()**

**{**

**Student s;**

**s.get();**

**return 0;**

**}**

**RESULT:**

**EXPERIMENT NO : 4.b**  **DATE :**

**SUM OF PRICES**

**AIM:**

**ALGORITHM**

1. **Start**
2. **Define Class: Item**
3. **Public Variables:**
   1. int price[10] (array to hold prices of 10 products)
   2. char product[10][20] (2D array to hold names of 10 products, each up to 20 characters)
4. **Public Methods:**
   1. void info() (to input product information)
   2. void display(int sum) (to display the total price)
5. **Implement Method: info()**
6. Declare variable int sum = 0 to store the total price.
7. **Implement Method: display(int sum)**
8. Output the total price of the products.
9. **Define Main Function**
10. **Stop**

**SUM OF PRICE**

**#include<iostream>**

**using namespace std;**

**class item**

**{**

**public:**

**int price[10];**

**char product[10];**

**void info();**

**void display(int sum);**

**};**

**void info()**

**{**

**int sum=0,i;**

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

**OUTPUT:**

**Enter product name: Apple**

**Enter the price: 30**

**Enter product name: Banana**

**Enter the price: 20**

**Enter product name: Orange**

**Enter the price: 25**

**Enter product name: Mango**

**Enter the price: 40**

**Enter product name: Grapes**

**Enter the price: 50**

**Enter product name: Kiwi**

**Enter the price: 35**

**Enter product name: Peach**

**Enter the price: 45**

**Enter product name: Plum**

**Enter the price: 55**

**Enter product name: Cherry**

**Enter the price: 60**

**Enter product name: Pomegranate**

**Enter the price: 70**

**Sum of product's prices: 500**

**{**

**cout<<"Enter product name:";**

**cin>>product;**

**cout<<"Enter the price:";**

**cin>>price;**

**}**

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

**{**

**sum=sum+price[i];**

**}**

**display(sum);**

**}**

**void display(int sum)**

**{**

**cout<<"Sum of product's prices:"<<sum;**

**}**

**int main()**

**{**

**item i;**

**i.info();**

**return 0;**

**}**

**EXPERIMENT NO :5.a**  **DATE :**

**AIM:**

**CONSTRUCTOR AND DESTROCTOR**

**AREA OF RECTANGLE**

* **Start**
* **Define Class: Area**
* **Private Variables:**
  + int l (length)
  + int b (breadth)
* **Public Methods:**
  + Area() (constructor)
  + void area\_rectangle(int length, int breadth) (to calculate and display area)
* **Implement Constructor: Area()**
* **Implement Method: area\_rectangle(int length, int breadth)**
* **Define Main Function**
* **Stop**

**#include<iostream>**

**using namespace std;**

**class area**

**{**

**int l,b;**

**public:**

**area()**

**{**

**int l,b;**

**}**

**void area\_rectangle(int l,int b)**

**{**

**int A;**

**A=l\*b;**

**cout<<"Area of rectangle:"<<A;**

**OUTPUT**

**Enter length and breadth: 5 10**

**Area of rectangle: 50**

**}**

**};**

**int main()**

**{**

**int l,b;**

**area a;**

**cout<<"Enter length and breadth:";**

**cin>>l;**

**cin>>b;**

**a.area\_rectangle(l,b);**

**return 0;**

**}**

**EXPERIMENT NO : 5.bDATE :**

**AIM:**

**ALGORITHM**

1. **Start**
2. **Define Class: Factorial**
3. **Private Variables:**
   1. int f (to store the factorial result)
4. **Public Methods:**
   1. Factorial() (constructor to initialize f)
   2. void fact() (to calculate the factorial)
5. **Implement Constructor: Factorial()**
6. **Implement Method: fact()**
7. **Define Main Function**l
8. **Stop**

**FACTORIAL**

**#include<iostream>**

**using namespace std;**

**class factorial**

**{**

**int f=1,n;**

**public:**

**factorial()**

**{**

**int f=1,n;**

**}**

**void fact(int n)**

**{**

**int i;**

**cout<<"Enter a number:";**

**cin>>n;**

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

**{**

**OUTPUT:**

**Enter a number: 5**

**Factorial = 120**

**f=f\*i;**

**}**

**cout<<"Factorial="<<f;**

**}**

**};**

**int main()**

**{**

**int n;**

**factorial facto;**

**facto.fact(n);**

**return 0;**

**}**

**EXPERIMENT NO : 5.c**  **DATE :**

**AIM:**

**ALGORITHM**

1. **Start**
2. **Define Class: fibonacci**
3. **Members:**
   1. int first: To store the first number in the Fibonacci series (initialized to 0).
   2. int second: To store the second number in the Fibonacci series (initialized to 1).
   3. int next: To store the next number in the Fibonacci series.
4. **Constructor:**
   1. fibonacci(): Initializes first to 0, second to 1, and next to 0.
5. **Method: void fibonacci\_series(int n)**
6. **Define Main Function**
7. **End**

**FIBONACCI**

**#include<iostream>**

**using namespace std;**

**class fibonacci**

**{**

**int first,second,next,n;**

**public:**

**fibonacci()**

**{**

**first=0;**

**second=1;**

**next=0;**

**}**

**void fibonacci\_series(int n)**

**{**

**int i;**

**cout<<"Enter a limit to print fibonacci series:";**

**Output:**

**Enter a limit to print Fibonacci series: 10**

**0 1 1 2 3 5 8 13 21 34**

**cin>>n;**

**cout<<first<<" ";**

**cout<<second<<" ";**

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

**{**

**next=first+second;**

**cout<<next<<" ";**

**first=second;**

**second=next;**

**}**

**}**

**};**

**int main()**

**{**

**int n;**

**fibonacci f;**

**f.fibonacci\_series(n);**

**return 0;**

**}**

**EXPERIMENT NO : 5.d**  **DATE :**

**AIM:**

1. **Start**
2. **Define Class: Sum**
3. **Private Variables:**
   1. int a[10] (to store the elements of the array)
   2. int sum (to hold the sum of the array elements)
4. **Public Methods:**
   1. Sum() (constructor to initialize sum)
   2. void Sum\_of\_array() (to calculate the sum of the array)
5. **Implement Constructor: Sum()**
6. Initialize sum to 0.
7. **Implement Method: Sum\_of\_array()**
8. **Define Main Function**
9. **Stop**

**SUM OF ARRAY**

**#include<iostream>**

**using namespace std;**

**class Sum**

**{**

**int a[10],sum=0;**

**public:**

**Sum()**

**{**

**int a[10],n,sum=0;**

**}**

**void Sum\_of\_array(int n)**

**{**

**int i;**

**cout<<"Enter number of elements:";**

**cin>>n;**

**cout<<"Enter array elements:";**

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

**OUTPUT:**

**Enter number of elements (max 10): 5**

**Enter array elements:**

**10**

**20**

**30**

**40**

**50**

**Given array:**

**10 20 30 40 50**

**SUM = 150**

**{**

**cin>>a[i];**

**}**

**cout<<"Given array:";**

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

**{**

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

**}**

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

**{**

**sum=sum+a[i];**

**}**

**cout<<"SUM="<<sum;**

**}**

**};**

**int main()**

**{**

**int n;**

**Sum s;**

**s.Sum\_of\_array(n);**

**}**

**EXPERIMENT NO :5.e DATE :**

**AIM:**

* **Start**
* **Define Class: Complex**
* **Members:**
  + int real: To store the real part of the complex number.
  + int imaginary: To store the imaginary part of the complex number.
* **Constructor:**
  + Complex(int tempReal = 0, int tempImaginary = 0): Initializes real and imaginary parts with the given values (default to 0).
* **Method: Complex addComp(Complex C2)**
  + Create a temporary Complex object temp.
* **Define Main Function**
* **End**

**ADDITION OF TWO COMPLEX NUMBERS**

**#include<iostream>**

**using namespace std;**

**class Complex**

**{**

**public: int real, imaginary;**

**Complex(int tempReal = 0,**

**int tempImaginary = 0)**

**{**

**real = tempReal;**

**imaginary = tempImaginary;**

**}**

**Complex addComp(Complex C1, Complex C2)**

**{**

**Complex temp;**

**temp.real = C1.real + C2.real;**

**temp.imaginary = (C1.imaginary + C2.imaginary);**

**return temp;**

**}**

**OUTPUT:**

**Complex number 1: 3 + i2**

**Complex number 2: 9 + i5**

**Sum of complex numbers: 12 + i7**

**};**

**int main()**

**{**

**Complex C1(3, 2);**

**cout << "Complex number 1 : " <<**

**C1.real << " + i" <<**

**C1.imaginary << endl;**

**Complex C2(9, 5);**

**cout << "Complex number 2 : " <<**

**C2.real << " + i" <<**

**C2.imaginary << endl;**

**Complex C3;**

**C3 = C3.addComp(C1, C2);**

**cout << "Sum of complex number : " <<**

**C3.real << " + i" <<**

**C3.imaginary;**

**}**

**EXPERIMENT NO : 5.f**  **DATE :**

**AIM:**

* **Start**
* **Define Class: distance**
* **Members:**
  + int feet: To store the feet part of the distance.
  + float inches: To store the inches part of the distance.
* **Constructor:**
  + distance(int d1 = 0, float d2 = 0.0): Initializes feet with d1 and inches with d2.
* **Method: void get()**
  + Prompt user to enter feet and inches.
  + Store the input values in feet and inches.
* **Method: void display()**
  + Print the distance in the format: feet' inches".
* **Define Main Function**
* **End**

**COPY CONSTRUCTOR**

**#include <iostream>**

**using namespace std;**

**class distance {**

**int feet;**

**float inches;**

**public:**

**distance(int d1 = 0, float d2 = 0.0) {**

**feet = d1;**

**inches = d2;**

**}**

**void get() {**

**cout << "Enter feet: ";**

**cin >> feet;**

**OUTPUT:**

**Distance d1: 11' 2.5"**

**Distance d2 (copy of d1): 11' 2.5"**

**Distance d3 (copy of d1): 11' 2.5"**

**cout << "Enter inches: ";**

**cin >> inches;**

**}**

**void display() {**

**cout << feet << "' " << inches << "\"" << endl;**

**}**

**};**

**int main() {**

**distance d1(11, 2.5);**

**distance d2(d1);**

**distance d3 = d1;**

**cout << "Distance d1: ";**

**d1.display();**

**cout << "Distance d2 (copy of d1): ";**

**d2.display();**

**cout << "Distance d3 (copy of d1): ";**

**d3.display();**

**return 0;**

**}**