#### Arithmetic Operations

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

int x, y, sum, diff, prod, quo;

cout << "Enter 2 numbers ";

cin >> x >> y;

sum = x+y; diff = x-y;

prod = x\*y; quo = x/y;

cout << "Sum is " << sum;

cout << "\nDifference is " << diff;

cout << "\nProduct is " << prod;

cout << "\nQuotient is " << quo;

getch();

}

#### Read & Write Values of different Data Types

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

int x; char a[10]; float f;

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

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

cout << "Enter a decimal value: "; cin >> f;

cout << "\nEntered values are:\n";

cout << x <<"\t";

cout << a <<"\t";

cout << f;

getch();

}

#### Swapping Values

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

int x, y, t;

cout << "Enter 2 numbers X & Y: ";

cin >> x >> y;

t = x; x = y; y = t;

cout << "X is now " << x;

cout << "\nY is now " << y;

getch();

}

|  |  |
| --- | --- |
| Star Pattern #include <iostream>  #include <conio.h>  using namespace std;  int main(){  int i, j;  for (i = 0; i <= 5; i++){  for (j = 0; j < i; j++){  cout << "\*";  }  cout << "\n";  }  getch();  }  \*  \*\*  \*\*\*  \*\*\*\*  \*\*\*\*\* | Character Pattern #include <iostream>  #include <conio.h>  using namespace std;  int main(){  int i, j;  for (i = 0; i <= 5; i++){  for (j = 0; j < i; j++){  cout << ((char) (j+65));  }  cout << "\n";  }  getch();  }  A  AB  ABC  ABCD  ABCDE |
| Leap Year Checker #include <iostream>  #include <conio.h>  using namespace std;  int main(){  int yr, i;  while(1){  cout << "\nEnter a year ";  cin >> yr;  if (( yr%4 == 0 ) && ( yr%100 != 0 ))  cout << "This is a Leap Year";  else if (( yr%100 == 0 ) && ( yr%400 == 0))  cout << "This is a Leap Year";  else if ( yr%400 == 0 )  cout << "This is a Leap Year";  else  cout << "This is not a Leap Year";  }  getch();  } | Prime or Not Prime #include <iostream>  #include <conio.h>  using namespace std;  int main(){  while(1){  int i,n,f = 0;  cout << "\nEnter a number: ";  cin >> n;  for(i = 2; i <= n; i++){  if(n%i == 0){  f = 1;  break;  }  }  if(f == 1)  cout << "Not Prime";  else  cout << "Prime";  }  getch();  } |
| Design a class employee with data members & member functions by using the scope resolution operator. #include <iostream>  #include <conio.h>  using namespace std;  class employee{  int emp\_no;  char name[16];  public: void read(); void write();  };  void employee::read(){  cout << "Enter Emp No & Name: ";  cin >> emp\_no >> name;  }  void employee::write(){  cout << emp\_no << " " << name;  }  int main(){  employee e1;  e1.read(); e1.write();  getch();  } | Design a class employee with data members & member functions w/o using the scope resolution operator. #include <iostream>  #include <conio.h>  using namespace std;  class employee{  int emp\_no;  char name[16];  public: void read(){  cout << "Enter Emp No & Name: ";  cin >> emp\_no >> name;  }  void write(){  cout << emp\_no << " " << name;  }  };  int main(){  employee e1;  e1.read(); e1.write();  getch();  } |

#### Constructors & De-constructor

#include <iostream>

#include <conio.h>

using namespace std;

class money{

int rs, paisa;

public :

money();

money(int r, int p);

money (money &m);

~money();

void read(); void show();

};

money::money(){

rs = paisa = 0;

}

money::money(int r, int p){

rs = r; paisa = p;

}

money::money(money &m){

rs = m.rs; paisa = m.paisa;

}

void money::read(){

cin >> rs >> paisa;

}

void money::show(){

cout << rs << " " << paisa;

}

money::~money(){ }

int main(){

money m1, m4;

cout << "1st amount: ";

m1.show();

money m2(100, 20);

cout << "\n2nd amount: ";

m2.show();

money m3(m2);

cout << "\n3rd amount: ";

m3.show();

cout << "\nEnter an amount: ";

m4.read(); m4.show();

getch();

}

#### Largest of 9 Numbers

#include <iostream>

#include <conio.h>

using namespace std;

int main(){

cout << "18BCAN024\n\n";

int i, e, f;

cout << "Enter 10 Numbers: ";

cin >> f;

for (i = 1; i < 10; i++){

cin >> e;

f = (e > f ? e : f);

}

cout << "Largest Number is " << f;

getch();

}

#### Inline Function

#include <iostream>

#include <conio.h>

using namespace std;

inline int sum(int a, int b){

return a + b;

}

inline int cube(int n){

return n \* n \* n;

}

int main(){

cout << "Sum is " << sum(12332, 78945);

cout << "\n";

cout << "Cube is " << cube(19);

getch();

}

#### Friend Function

#include <iostream>

#include <conio.h>

using namespace std;

class person;

class money{

int rs, paisa;

public: void read(){

cout << "Enter earnings: ";

cin >> rs >> paisa;

}

friend void display(person p, money m);

};

class person{

char name[16];

public: void input(){

cout << "Enter Name: "; cin >> name;

}

friend void display(person p, money m);

};

void display(person p, money m){

cout << p.name << " earns " << m.rs << "." << m.paisa << " rupees.";

}

int main(){

money m1; m1.read();

person p1; p1.input();

display(p1, m1);

getch();

}

#### Swap private data of 2 classes using friend function

#include <iostream>

#include <conio.h>

using namespace std;

class digit;

class number{

int a;

public: void read(){

cout << "Enter a number: ";

cin >> a;

}

void show(){

cout << "A = "<< a << "\n";

}

friend void swap(digit d, number n);

};

class digit{

int b;

public: void read(){

cout << "Enter a number: ";

cin >> b;

}

void show(){

cout << "B = "<< b << "\n";

}

friend void swap(digit d, number n);

};

void swap(digit d, number n){

int temp;

temp = n.a; n.a = d.b; d.b = temp;

cout << "\nAfter Swapping: \n";

cout << "A = " << n.a << "\n" << "B = " << d.b;

}

int main(){

number n1;

digit d1;

n1.read();

d1.read();

cout << "\nBefore swapping:\n";

n1.show();

d1.show();

swap(d1, n1);

getch();

}

|  |  |
| --- | --- |
| Unary operator (-) overloading #include <iostream>  #include <conio.h>  using namespace std;  class space{  int x, y;  public: void read(int a, int b){  x = a; y = b;  }  void display(){  cout << x << " " << y << "\n";  }  void operator -();  };  void space::operator -(){  x = -x;  y = -y;  }  int main(){  space s1;  s1.read(10, -20);  s1.display();  -s1;  s1.display();  getch();  } | Unary operator (-) overloading using Friend Function #include <iostream>  #include <conio.h>  using namespace std;  class space{  public: int x, y;  void read(int a, int b){  x = a; y = b;  }  void display(){  cout << x << " " << y << "\n";  }  void operator -(space &s);  };  void operator -(space &s){  s.x = -s.x;  s.y = -s.y;  }  int main(){  space s1;  s1.read(10, -20);  s1.display();  -s1;  s1.display();  getch();  } |
| Binary operator (+) overloading #include <iostream>  #include <conio.h>  using namespace std;  class Complex{  int x, y;  public:  void read(){  cout << "Enter 2 Numbers: ";  cin >> x >> y;  }  Complex operator +(Complex &c){  Complex tmp;  tmp.x = x + c.x;  tmp.y = y + c.y;  return tmp;  }  void display(){  cout << x << " " << y;  }  };  int main(){  Complex c1, c2, add;  c1.read();  c2.read();  add = c1 + c2;  cout << "Addition is: ";  add.display();  getch();  } | Binary operator (\*) overloading using Friend function #include <iostream>  #include <conio.h>  using namespace std;  class Complex{  int x, y;  public:  void read(){  cout << "Enter 2 Numbers: ";  cin >> x >> y;  }  void display(){  cout << x << " " << y;  }  friend Complex operator \*(Complex c1, Complex c2);  };  Complex operator \*(Complex c1, Complex c2){  Complex c;  c.x = c1.x \* c2.x;  c.y = c1.y \* c2.y;  return(c);  }  int main(){  Complex c1, c2, mul;  c1.read();  c2.read();  mul = c1 \* c2;  cout << "Multiplication is: ";  mul.display();  getch();  } |
| Single-level Inheritance #include <iostream>  #include <conio.h>  using namespace std;  class base{  public: int x;  void getdata(){  cout << "Enter X = ";  cin >> x;  }  };  class derive : public base{  private: int y;  public: void readdata(){  cout << "Enter Y = ";  cin >> y;  }  void sum(){  cout << " X + Y = " << x + y;  }  };  int main(){  derive a;  a.getdata();  a.readdata();  a.sum();  getch();  } | Multiple Inheritance #include <iostream>  #include <conio.h>  using namespace std;  class A{  protected: int a;  public: void get\_a(){  cout << "Enter A: ";  cin >> a;  }  };  class B{  protected: int b;  public: void get\_b(){  cout << "Enter B: ";  cin >> b;  }  };  class C : public A, public B{  public: void display(){  cout << "A = " << a << "\n";  cout << "B = " << b << "\n";  cout << "A + B = "<< a + b;  }  };  int main(){  C c;  c.get\_a();  c.get\_b();  c.display();  getch();  } |
| File handling - add values to file #include <iostream>  #include <fstream>  using namespace std;  int main(){  fstream file;  file.open("abc.txt");  int i, val;  for(i = 0; i < 10; i++){  cout << "Enter a value: ";  cin >> val;  file << val << "\n";  }  file.close();  } | File handling - read values from file #include <iostream>  #include <fstream>  using namespace std;  int main(){  fstream file;  file.open("abc.txt");  int i, val;  for(i = 0; i < 10; i++){  file >> val;  cout << val << "\n";  }  file.close();  } |