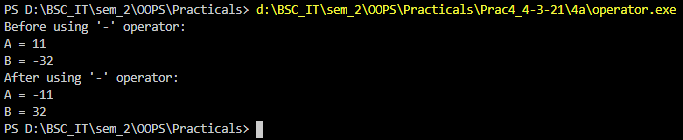
Practical 4: Operator Overloading

1. Overload the operator unary (-) for demonstrating operator overloading.

Code:

|  |
| --- |
| // Overload the operator unary(-) for demonstrating operator overloading  #include<iostream>  class optr  {  private:      int a, b;  public:      optr(int x, int y){ a=x, b=y; }      void operator-(){          a=-a, b=-b;      }      void print(){          std::cout << "A = " << a << std::endl;          std::cout << "B = " << b << std::endl;      }      ~optr(){}  };  int main(){      optr one(11,-32);      std::cout << "Before using '-' operator:" << std::endl;      one.print();      -one;      std::cout << "After using '-' operator:" << std::endl;      one.print();      return 0;  } |

Output:

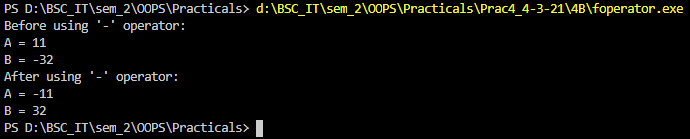


1. Overload the operator unary(-) for demonstrating operator overloading using friend function.

Code:

|  |
| --- |
| // Overload the operator unary(-) for demonstrating operator overloading using friend function.  #include<iostream>  class optr  {  private:      int a, b;  public:      optr(){ a=0, b=0; }      optr(int x, int y){ a=x, b=y; }      friend void operator-(optr &in);      void print(){          std::cout << "A = " << a << std::endl;          std::cout << "B = " << b << std::endl;      }      ~optr(){}  };  void operator-(optr &in){          in.a = -in.a, in.b = -in.b;  }  int main(){      optr one(11,-32);      std::cout << "Before using '-' operator:" << std::endl;      one.print();      -(one);      std::cout << "After using '-' operator:" << std::endl;      one.print();      return 0;  } |

Output:

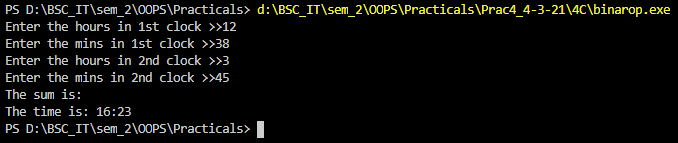


1. Overload the operator + for adding the timings of two clocks. And also pass objects as an argument.

Code:

|  |
| --- |
| /\*      Overload the operator + for adding      the timings of two clocks, And also      pass objects as an argument.  \*/  #include<iostream>  #include<math.h>  class clock  {  private:      float hours, minutes;  public:      clock(){};      clock(float x, float y){ hours = x; minutes = y; }      clock operator+(clock t1)      {          float temp;          clock result;          result.minutes = minutes + t1.minutes;          temp = result.minutes;          result.minutes = fmod(result.minutes, 60);          result.hours = hours + t1.hours + floor(temp/60);          return result;      }      void disp(){          std::cout << "The time is: " << hours <<":"<< minutes;      }      ~clock(){}  };  int main()  {      float x, y;      std::cout << "Enter the hours in 1st clock >>";      std::cin >> x;      std::cout << "Enter the mins in 1st clock >>";      std::cin >> y;      clock c1(x, y);      std::cout << "Enter the hours in 2nd clock >>";      std::cin >> x;      std::cout << "Enter the mins in 2nd clock >>";      std::cin >> y;      clock c2(x, y);      clock sum = c1 + c2;      std::cout << "The sum is: " << std::endl;      sum.disp();  } |

Output:

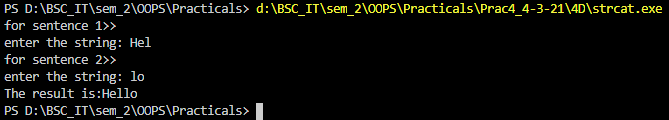


1. Overload the + for concatenating the two strings. For e.g “Py” +“thon” =Python

Code:

|  |
| --- |
| /\*      Overload the + for concatenating the two strings.      For e.g “Py” +“thon” =Python  \*/  #include<iostream>  #include<string.h>  using std::cout;  using std::cin;  class str  {  private:      char c1[30];  public:      str(){ }      void getstr(){          cout << "enter the string: ";          cin >> c1;      }      str operator+(str y)      {          str result;          strcpy(result.c1, c1);          strcat(result.c1, y.c1);          return result;      }      void disp(){          cout << c1;      }      ~str(){      }  };  int main()  {      str sent1, sent2, sent3;      cout << "for sentence 1>> \n";      sent1.getstr();      cout << "for sentence 2>> \n";      sent2.getstr();      sent3 = sent1+sent2;      cout<<"The result is:";      sent3.disp();      return 0;  } |

Output:



Write-up:

