Об’єктно-орієнтова не програмування

Завдання №2

Завдання 1

Створіть клас Numbers, який містить два цілих числа. Цей клас повинен мати дві змінні-члени для зберігання цих двох цілих чисел. Ви також повинні створити два методи:

- метод set(), який дозволить присвоювати значення змінним;
- метод print(), який буде виводити значення змінних.

Виконання наступної функції main():

```
1  int main()
2  {
3      Numbers n1;
4      n1.set(3, 3); // ініціалізуємо об'єкт n1 значеннями 3 і 3
```

```
5
6       Numbers n2{ 4, 4 }; // ініціалізуємо об'єкт n2 значеннями 4 і 4
7
8       n1.print();
9       n2.print();
10
11      return 0;
12  }
```

Повинно видавати наступний результат:

```
Numbers(3, 3)
Numbers(4, 4)
```

Завдання 2

Чому для Numbers повинен використовуватися клас, а не структура?

Завдання 3

а) Напишіть простий клас з іменем Numbers. Цей клас повинен мати:

- три закриті змінні-члени типу double: `m_a`, `m_b` і `m_c`;
- відкритий метод з іменем setValues(), який дозволить встановлювати значення для `m_a`, `m_b` і `m_c`;
- відкритий метод з іменем print(), який виводитиме об'єкт класу Numbers в наступному форматі:

<m_a, m_b, m_c>.

Наступний код функції main():

```
int main()
{
    Numbers point;
    point.setValues(3.0, 4.0, 5.0);

    point.print();

    return 0;
}
```

Повинен видавати наступний результат:

```
<3, 4, 5>
```

Завдання 4

b) Додайте функцію isEqual() в клас Numbers, щоб наступний код працював коректно:

```
int main()
{
    Numbers point1;
    point1.setValues(3.0, 4.0, 5.0);

    Numbers point2;
    point2.setValues(3.0, 4.0, 5.0);

    if (point1.isEqual(point2))
        std::cout << "point1 and point2 are equal\n";
    else
```

```
        std::cout << "point1 and point2 are not equal\n";

    Numbers point3;
    point3.setValues(7.0, 8.0, 9.0);

    if (point1.isEqual(point3))
        std::cout << "point1 and point3 are equal\n";
    else
        std::cout << "point1 and point3 are not equal\n";

    return 0;
}
```

Завдання 5

Тепер спробуємо щось складніше. Напишіть клас, який реалізує функціонал **стеку**. Клас Stack повинен містити:

- закритий цілочисельний **фіксований масив** довжиною 10 елементів;
- закрите цілочисельне значення для відслідковування довжини стеку;
- відкритий метод з іменем reset(), який ініціалізуватиме значенням `0` довжину і всі значення

елементів;

- відкритий метод з іменем push(), який додаватиме значення в стек. Метод push() повинен повертати значення `false`, якщо масив вже заповнений, в протилежному випадку — `true`;

- відкритий метод з іменем pop() для повернення значень зі стеку. Якщо в стеці немає значень, то повинен виводитись **стейтмент assert**;

- відкритий метод з іменем print(), який виводитиме всі значення стеку.

Наступний код функції main():

```
int main()
{
    Stack stack;
    stack.reset();

    stack.print();

    stack.push(3);
    stack.push(7);
    stack.push(5);
    stack.print();

    stack.pop();
    stack.print();
```

```
15
16      stack.pop();
17      stack.pop();
18
19      stack.print();
20
21      return 0;
22  }
```

Повинен видавати наступний результат:

```
( )
( 3 7 5 )
( 3 7 )
( )
```