

1. Поліморфізм - це ...

- ... кожна реалізація в іншій функції
- ... одна функція, з різними реалізаціями
- ... одна реалізація, з різними функціями

2. Заповніть пробіли, щоб оголосити клас "Enemy" із захищеною змінною "attackPower" та публічною функцією " setAttackPower ".

```
____ Enemy {  
____ :  
int attackPower;  
____ :  
void setAttackPower( __ a) {  
    attackPower = a;  
}  
};
```

3. Оголосіть об'єкти "Ninja" і "Monster", а потім два вказівники "Enemy", які вказують на об'єкт "Ninja" та "Monster", відповідно.

```
____ ninjaObj;  
____ monsterObj;  
Enemy* e1 = ____;  
____ e2 = &monsterObj;
```

4. Оголосіть вказівники Enemy на об'єкти "Ninja" та "Monster". Встановіть "attackPower" цих об'єктів за допомогою вказівників "Enemy", а потім викликайте функції attack () об'єктів "Ninja " і " Monster ":

```
Enemy _ e1 = &ninjaObj;  
Enemy* e2 = __ monsterObj;  
e1 __ setAttackPower(29);  
e2 __ setAttackPower(99);  
ninjaObj.attack();  
monsterObj. ____ ();
```

5. Оголосіть віртуальну функцію під назвою "attack ()" для класу "Enemy".

```
_____ Enemy {  
public:  
    _____ void attack() {  
        cout << "enemy attacks"; }  
};
```

6. Заповніть пробіли, щоб оголосити клас "Enemy" з віртуальною функцією атаки, а потім оголосити клас "Ninja", який успадковується від "Enemy" і переосмислює його віртуальну функцію атаки.

```
class Enemy {  
public:  
    _____ void attack() {  
        cout << "Enemy attacks"; }  
};  
_____ Ninja : _____ Enemy {  
public:  
    void _____ () {  
        cout << "Ninja attacks"; }  
};
```

7. Клас називається поліморфним, якщо ...

- ... це було оголошено за допомогою ключового слова "polymorphic"
- ... він має дружню функцію
- ... він має віртуальну функцію

8. При виклику віртуальної функції за допомогою вказівника базового класу ...

- ... вона викликає функцію базового класу
- ... вона нічого не робить
- ... вона викликає ту функцію класу, на яку вказує вказівник базового класу

9. Оголосіть чисту віртуальну функцію "foo ()"

```
_____ void foo() _ 0;
```

10. Чиста віртуальна функція ...

- ... не має тіла і має бути реалізована у похідних класах
- ... повинна мати реалізацію
- ... завжди повинна повернути void

11. Заповніть пробіли, щоб оголосити клас Person чистою віртуальною функцією sayHello () та переписіть її у похідному класі Student.

```
class Person {  
public:  
    _____ void sayHello() _ 0;  
};  
class Student : _____ Person {  
public:  
    _____ sayHello() {  
        cout << "Student says hello"; }  
};
```

12. Абстрактний клас - це клас, який ...

- ... було оголошено за допомогою ключового слова “abstract”
- ... має щонайменше два оголошених метода
- ... має чисту віртуальну функцію