Блиц тест №4 - ООП практикум
17.04.2024

1. **Защо не е препоръчително да се хвърлят грешки в деструктора? Обосновете се и дайте пример.**

2. **Какво ще се отпечата на конзолата?**

```
struct A
{
    A()
    {
        static unsigned count = 0;

        std::cout << "A()" << std::endl;
        if (count == 3)
        {
            throw std::exception("Error!");
        }

        count++;
    }

    ~A()
    {
        std::cout << "~A()" << std::endl;
    }
};

class X
{
    A a;
    B b;
    C c;

public:
    X()
    {
        std::cout << "X()" << std::endl;
    }

    ~X()
    {
        std::cout << "~X()" << std::endl;
    }
};

constexpr int MAX_SIZE = 10;

int main()
{
    try
    {
        X arr[10];
    }
    catch (std::exception& ex)
    {
        std::cout << ex.what() << std::endl;
    }
}
```

```
struct B
{
    B()
    {
        std::cout << "B()" << std::endl;
    }

    ~B()
    {
        std::cout << "~B()" << std::endl;
    }
};
```

```
struct C
{
    C()
    {
        std::cout << "C()" << std::endl;
    }

    ~C()
    {
        std::cout << "~C()" << std::endl;
    }
};
```

3. **Нека приемем, че съществува функция f() такава, че в нея има възможност за хвърляне на следните exception-и: std::exception, std::runtime_error и std::range_error. Разпишете try-catch блок за функцията f().**

4. **Допусната ли е грешка в следния фрагмент код? Обосновете отговора си.**

```
namespace Weekdays
{
    constexpr unsigned short MONDAY = 1;
    constexpr unsigned short TUESDAY = 2;
    constexpr unsigned short WEDNESDAY = 3;
    constexpr unsigned short THURSDAY = 4;
    constexpr unsigned short FRIDAY = 5;
    constexpr unsigned short SATURDAY = 6;
    constexpr unsigned short SUNDAY = 7;
}


class Weekday
{
    unsigned short day;

public:
    Weekday(unsigned short day)
    {
        this->day = day;
    }

    void printWeekday() noexcept
    {
        switch (day)
        {
        case Weekdays::MONDAY:
            std::cout << "Monday" << std::endl;
            break;
        case Weekdays::TUESDAY:
            std::cout << "Tuesday" << std::endl;
        case Weekdays::WEDNESDAY:
            std::cout << "Wednesday" << std::endl;
            break;
        case Weekdays::THURSDAY:
            std::cout << "Thursday" << std::endl;
        case Weekdays::FRIDAY:
            std::cout << "Friday" << std::endl;
            break;
        case Weekdays::SATURDAY:
            std::cout << "Saturday" << std::endl;
        case Weekdays::SUNDAY:
            std::cout << "Sunday" << std::endl;
        default:
            throw std::logic_error("Weekday must be from 1 to 7.");
        }
    }
};

int main()
{
    Weekday day(3);
    day.printWeekday();
}
```

5. **Кои от следните оператори можем да предефинираме?**
    a. :: (scope resolution)
    b. && (and)
    c. sizeof
    d. () (function call)
6. **Вярно ли е, че когато предефинираме оператор, можем да променим броя на неговите параметри?**
7. **Разгледайте по-долу разписания фрагмент. Ще се компилира ли успешно той?**

```
class Test
{
    void operator[](std::string x) {}
};
```

a. Да
b. Не, защото operator[] е private функция
c. Не, защото operator[] не е void
d. Не, защото параметърът на operator[] е std::string, а трябва да бъде std::size_t

8. Нека е дадена следната дефиниция:

class A { public: int arr[10];};
Да приемем, че желаем да дефинираме операторна функция, такава че след изпълнението на оператора a2 = 3 + a1; да е изпълнено a2.arr[i] == 3 + a1.arr[i], за всяко i от 0 до 9, където a1 и a2 са обкет и от клас A. Как трябва да се реализира оператора + ?

a. Вътрешен операторен метод със сигнатура void operator+(int, const A&)
b. Вътрешен операторен метод със сигнатура A operator+(int, const A&)
c. Външен за класа оператор void operator+(int, const A&)
d. Външен за класа оператор A& operator+(int, const A&)
e. Външен за класа оператор A operator+(int, const A&)