Практическое занятие 3.18

Повторение: наследование, виртуальные члены класса

placement new

Вопросы на повторение

1. **Можно** ли у класса не описывать конструктор? А у класса наследника?
2. **Может** ли конструктор быть ***protected*** или ***private***? А деструктор?
3. **Зачем** нужен виртуальный деструктор?
4. **Может** ли конструктор быть виртуальным?
5. **Может** ли конструктор выбрасывать исключения? А деструктор?

Вопросы на повторение

6. **Как** защитить объект от копирования?
7. **Как** запретить создавать объект на стеке?
8. **Для** чего используется вызов ***throw*** без аргументов?
9. **Что** происходит при присваивании ***unique_ptr***?

Вопросы на повторение

10. Что “видит” processWidget в этом коде?

```
class Widget {…};

class SpecialWidget : public Widget {…};

void processWidget(Widget w);

...

SpecialWidget sw;
processWidget(sw);  // ..?
```

Вопросы на повторение

11. В чем опасность такого метода?

```
int* function() {
    int *x = new int(2);
    return x;
}
```

12. Для чего нужны деструктор и оператор **delete**?

Вопросы на повторение

13. Какие функции-члены класса автоматически создаются при объявлении ”пустого”, например class A { }; ?

14. Допустимо ли комбинировать в объявлении метода класса:

- **static virtual**?
- **virtual const**?
- **static const**?

placement new

```
int* placementMemory = new int[100];

int* newArray1 = new (placementMemory) int[10];

Derived* derived = new (placementMemory + 20) Derived();

//delete derived;

derived->~Derived();

delete[] placementMemory;
```

operator new(), operator delete()

```
template <class T>
void func() {
     T* object = new T();
//…
     delete object;
}
```

Операции, выполняемые для new:

1. **Выделение памяти void***
2. Преобразование типа указателя к T*
3. Вызов конструктора

Код, выполняемый для delete:

1. Вызов деструктора
2. **Очистка памяти**