|  |  |
| --- | --- |
| Институт (факультет) | Институт информационных технологий |
| Кафедра | Кафедра математического и программного обеспечения ЭВМ |

**СОЗДАНИЕ И УНИЧТОЖЕНИЕ ОБЪЕКТОВ КЛАССА**

|  |  |
| --- | --- |
| **Дисциплина:** | ООП |
| **Темы:** | Конструкторы и деструкторы класса |

**Среда разработки:** Microsoft Visual Studio

**Язык программирования:** C++

**Тип проекта:** Консольное приложение

**Задание на лабораторную работу №2**

**ТРЕБОВАНИЯ К РАЗРАБОТКЕ**

1. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
2. Придерживайтесь принципа DRY (Don’t repeat yourself).
3. Обязательно наличие комментариев.

**ЦЕЛЬ РАБОТЫ**

Изучить синтаксис и семантику определения и вызова конструкторов и деструкторов. Приобрести практические навыки создания и уничтожения объектов. Изучить особенности применения различных видов конструкторов.

**ЗАДАНИЯ**

*4 часа*

1. Ознакомьтесь с УМП по ООП часть 1 раздел V.
2. Модифицируйте абстрактные типы данных, реализованные по заданию ЛР1, заменив соответствующие компонентные функции на систему конструкторов и деструктор.
3. Проверьте работоспособность АТД на тестовом наборе данных.

**ПРИМЕЧАНИЯ**

* При наличии компонентных данных типа указатель и их полной инициализации внутри класса, необходимо обеспечить корректную передачу объектов самого класса по значению в какую-либо функцию.
* Как мы знаем, передавая объект нашего класса в функцию по значению, в памяти создается его копия, и область видимости этой копии ограничена этой функцией. Как только функция будет выполнена, копия объекта нашего класса будет высвобождена из памяти.
* Конструктор по умолчанию, как и деструктор при компиляции будут создаваться автоматически, даже если их и не объявлять, вспоминаем первую лабораторную работу.
* Копия объекта класса, не имеющего в объявлении конструкторов, полностью копирует значения компонентных данных.
* Из этого всего следует, что требуется указать как правильно создавать копию объекта нашего класса, во избежание потери данных типа указатель. В противном случае, копия указателя будет инициализирована тем же адресом, что и у оригинального объекта. Уничтожая копию объекта, мы уничтожим так же и данные, на которые ссылаются указатели у оригинального объекта.

P.S. если вы будете видеть «кракозябры» у ваших строковых данных, после вызова какой-либо функции, где вы передаете объект класса по значению, как явно так и не явно, знайте, что произошло именно то, что было описано выше.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Каковы особенности синтаксиса и семантики конструкторов и деструкторов?

2. В чем заключаются особенности применения конструкторов по умолчанию?

3. В каких случаях вызывается копирующий конструктор?

4. При каких условиях конструктор может использоваться для преобразования типа?

5. В каких ситуациях необходимы инициализаторы конструктора, синтаксис и семантика их использования?

**Приложение 1**

**Варианты:**

1. Вычислительные машины
2. Строительные материалы
3. Строительные конструкции
4. Электроинструмент
5. Железнодорожный транспорт
6. Автомобильный транспорт
7. Воздушный транспорт
8. Водный транспорт
9. Энергетические установки
10. Запоминающие устройства
11. Устройства для управления компьютером
12. Телефонные аппараты
13. Бытовая техника
14. Устройства передачи данных
15. Программное обеспечение вычислительной техники
16. Печатающие устройства
17. Офисная мебель
18. Аппаратное обеспечение вычислительной техники
19. Элементная база для сборки компьютеров
20. Аппаратное обеспечение компьютерных сетей
21. Звуковоспроизводящая аппаратура
22. Звукозаписывающая аппаратура
23. Осветительные приборы
24. Оптические приборы
25. Нагревательные приборы
26. Холодильная техника
27. Электрические машины
28. Строительные машины
29. Металлургические машины
30. Сельскохозяйственные машины
31. Элементы интерьера
32. Канцелярские товары
33. Изделия целлюлозно-бумажной промышленности
34. Металлургические технологии
35. Врачебный контроль физиологического состояния спортсмена
36. Печи
37. Камины
38. Инструментальные среды для разработки программного обеспечения
39. Видеомониторы
40. Принтеры
41. Подъемно-транспортные машины
42. Плоттеры

**Приложение 2**

**Варианты:**

1. Статический вектор\*

2. Динамический вектор\*

3. Статическая матрица\*\*

4. Динамическая матрица\*\*

5. Линейный односвязный список\*

6. Линейный двусвязный список\*

7. Стек\*

8. Очередь\*

9. Закольцованный список\*

10. Двоичное дерево\*

\* - обязательно с использованием структуры и указателей.

\*\* - двумерный динамический массив.