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

**ПЕРЕГРУЗКА ОПЕРАТОРОВ**

|  |  |
| --- | --- |
| **Дисциплина:** | ООП |
| **Темы:** | Операторы |

**Среда разработки:** Microsoft Visual Studio

**Язык программирования:** C++

**Тип проекта:** Консольное приложение

**Задание на лабораторную работу №3**

**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 понятие перегрузки, приобрести практические навыки перегрузки отдельных операторов.

**ЗАДАНИЯ**

*4 часа*

1. Ознакомьтесь с УМП по ООП часть 2 раздел VI.
2. Модифицируйте абстрактные типы данных, реализованные по заданию ЛР2, перегрузив минимум 3 ключевых оператора.
3. Проверьте работоспособность АТД на тестовом наборе данных.

**ПРИМЕЧАНИЯ**

* Обязательно перегрузите оператор вывода на консоль и один и операторов сравнения у АТД.
* Используйте эти операторы в классе-контейнере, в первую очереди в функции сортировки и вывода элементов АТД.

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. **Назовите алгоритм соответствия для каждого аргумента.**
2. **С какой целью применяется перегрузка операторов?**

Перегрузка операторов в языке программирования C++ позволяет определить пользовательское поведение для стандартных операторов, когда они применяются к объектам пользовательских классов. Основная цель перегрузки операторов заключается в обеспечении более естественного и удобного интерфейса для работы с объектами, а также в создании более интуитивно понятного кода.

1. **Для каких операторов нельзя использовать перегрузку?**
2. **Оператор вызова функции (()):**
   * Оператор **()** для вызова функции не может быть перегружен.
3. **Оператор массива ([]):**
   * Оператор **[]** для работы с массивами не поддается перегрузке.
4. **Оператор размера типа (sizeof):**
   * Оператор **sizeof** не может быть перегружен.
5. **Оператор условного выражения (?:):**
   * Тернарный оператор **?:** не поддается перегрузке.
6. **Оператор точки в шаблонных функциях (.):**
   * Оператор **.** не перегружается для шаблонных функций.
7. **Оператор разрешения области видимости (::):**
   * Оператор **::** для разрешения области видимости не может быть перегружен.
8. **Оператор изменения приоритета выражения ((), [,]):**
   * Операторы, используемые для изменения приоритета выражений, такие как круглые скобки **()** и квадратные скобки **[,]**, не подлежат перегрузке.
9. **Оператор typeid в качестве выражения:**

* Оператор **typeid** в виде выражения (не как оператор **typeid(type)**) также не может быть перегружен.

1. **Операторы управления выделением памяти (new, delete, new[], delete[]):**

* Операторы **new**, **delete**, **new[]**, **delete[]** не могут быть перегружены.

1. **В каком случае используется функция-член, имеющая пустой список аргументов?**

Функция-член, имеющая пустой список аргументов (без параметров), называется **нульарной функцией-член**. Она может использоваться в следующих случаях:

1. **Доступ к членам данных без аргументов:**
   * Нульарные функции-члены могут использоваться для предоставления доступа к членам данных объекта без необходимости передачи аргументов. Например, функция-член, которая возвращает значение определенного члена данных объекта.
2. **Вызов методов, не требующих аргументов:**
   * Если метод объекта не требует передачи аргументов и выполняет какое-то действие, он может быть реализован как нульарная функция-член.
3. **Что произойдет, если перегруженная операция [ ] будет иметь тип возвращаемого значения int, а не int&?**

Если перегруженная операция [] возвращает int вместо int& (ссылка на int), это может привести к неожиданным результатам при использовании этой операции.

Когда [] возвращает int, то при использовании этой операции для доступа к элементу массива, возвращается копия значения элемента, а не ссылка на оригинальный элемент. Это означает, что изменения, внесенные в возвращенное значение, не будут отражаться на самом массиве. Это может привести к трудноуловимым ошибкам и нежелательному поведению в коде.

**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.