Лабораторная работа № 9

Цель работы

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, обрабатывающей исключительные ситуации.

Постановка задачи

1. Реализовать класс, перегрузить для него операции, указанные в варианте.
2. Определить исключительные ситуации.
3. Предусмотреть генерацию исключительных ситуаций.

> Класс- контейнер МНОЖЕСТВО с элементами типа int.
>
> Реализовать операции:
>
> [] – доступа по индексу;
>
> != - проверка на неравенство;
>
> < число – принадлежность числа множеству;
>
> +n – переход вправо к элементу с номером n .

Описание класса

Класс Array представляет собой контейнер для хранения массива целых чисел с фиксированным максимальным размером (MAX_SIZE = 30). Основные характеристики:

- Динамическое выделение памяти под массив
- Контроль границ массива
- Обработка исключительных ситуаций
- Перегрузка операторов для удобной работы с массивом

Определение компонентных функций

Основные методы класса:

- Конструкторы:
    - Array() - создает пустой массив
    - Array(int s) - создает массив заданного размера, инициализированный нулями
    - Array(int s, const int *mas) - создает массив из существующего массива
    - Array(const Array &v) - конструктор копирования
- Операторы:
    - operator= - присваивание массивов

    - operator[] - доступ к элементу по индексу
    - operator+ - добавление элемента в конец массива
    - operator-- - удаление последнего элемента
    - operator<< и operator>> - ввод/вывод

- Деструктор:
    - Освобождает выделенную память

Определение глобальных функций

Глобальные перегруженные операторы:

- ostream &operator<<(ostream &out, const Array &v) - вывод массива в поток
- istream &operator>>(istream &in, Array &v) - ввод массива из потока

Функция main()

```
int main() {
    try {
        Array x(2);
        Array y;

        cout << x;
        cout << "Номер элемента: ";
        int i;
        cin >> i;
        cout << x[i] << endl;

        y = x + 3;
        cout << y;

        --x;
        cout << x;
        --x;
        cout << x;
        --x;
    } catch (int) {
        cout << "Ошибка" << endl;
    }
    return 0;
}
```

Объяснение результатов работы программы

Программа демонстрирует:

1. Создание массива размером 2 (Array x(2))
2. Вывод массива (все элементы 0)
3. Доступ к элементу по индексу (с проверкой границ)

4. Добавление элемента в массив (x + 3)
5. Последовательное удаление элементов (--x)
6. Обработку исключений при:
    - Выходе за границы массива
    - Попытке добавить элемент в полный массив
    - Попытке удалить элемент из пустого массива

Ответы на контрольные вопросы

1. Для чего используется механизм наследования?

Ответ: Механизм наследования позволяет создавать новые классы на основе существующих, переиспользуя их функциональность. Это обеспечивает:

- Повторное использование кода
- Создание иерархии классов
- Полиморфное поведение объектов

2. Каким образом наследуются компоненты класса, описанные со спецификатором public?

Ответ: Компоненты с модификатором public:

- В public-наследовании остаются public
- В protected-наследовании становятся protected
- В private-наследовании становятся private

3. Каким образом наследуются компоненты класса, описанные со спецификатором private?

Ответ: Приватные компоненты не наследуются напрямую в производных классах, независимо от типа наследования. Однако они остаются частью объекта базового класса и могут быть доступны через public/protected методы базового класса.