Лабораторная работа №4

Декомпозиция текстового задания и выделение классов

Теоретический материал по std::vector

std::vector — это шаблонный класс из стандартной библиотеки C++, представляющий собой динамический массив. Он предоставляет множество методов для удобной работы с последовательностями элементов.

Данный контейнер, является наиболее частым способом организации данных в программах, написанных на C++. В рабочих задачах организация длительного хранения обеспечивается именно этим классом.

Класс является шаблонным – это означает, что внутренний тип данных может быть любым (примитивы, описанные вами классы, указатели). Содержимое определяется на этапе компиляции. Более подробно работа с шаблонами будет рассмотрена в дальнейшем.

1. **Конструкторы и деструктор**

- vector<T> v; – создаёт пустой вектор. *(T – любой тип данных)*
- vector<T> v(n); – вектор из n элементов (значение по умолчанию для типа T).
- vector<T> v(n, value); – вектор из n элементов, каждый равен value.
- vector<T> v(begin, end); – вектор, инициализированный элементами из диапазона [begin, end).
- vector<T> v(initializer_list); – вектор с элементами из списка инициализации {1, 2, 3}.
- ~vector() – автоматически освобождает память.

2. **Доступ к элементам**

- v[i] – доступ к элементу по индексу **без проверки границ** (аналог *(v.begin() + i)). (*Здесь и далее v – объект класса vector*)
- v.at(i) – доступ с **проверкой границ** (бросает std::out_of_range).
- v.front() – первый элемент.
- v.back() – последний элемент.
- v.data() – указатель на массив данных (как у C-массива).

3. **Итераторы (*Упрощенно – указатель*)**

- begin(), end() – итераторы на начало и конец.
- rbegin(), rend() – реверсивные итераторы.
- cbegin(), cend() – константные итераторы.

4. **Размер и ёмкость**

- v.size() – текущее количество элементов.
- v.empty() – true, если вектор пуст.
- v.capacity() – текущая вместимость (сколько элементов можно добавить без реаллокации).
- v.reserve(n) – резервирует память под n элементов (увеличивает capacity).
- v.shrink_to_fit() – уменьшает capacity до size() (не гарантировано).

5. **Изменение содержимого**

- v.push_back(x) – добавляет элемент x в конец.
- v.pop_back() – удаляет последний элемент.
- v.insert(pos, x) – вставляет x перед позицией pos.
- v.erase(pos) – удаляет элемент на позиции pos.
- v.erase(begin, end) – удаляет элементы в диапазоне [begin, end).
- v.clear() – очищает вектор (размер = 0, память может не освобождаться).
- v.resize(n) – изменяет размер (если n > size(), добавляются элементы по умолчанию).
- v.resize(n, value) – аналогично, но новые элементы инициализируются value.
- v.assign(n, value) – заменяет содержимое на n копий value.
- v.assign(begin, end) – заменяет содержимое элементами из диапазона.

6. **Обмен и сравнение**

- v1.swap(v2) – обменивает содержимое двух векторов.
- std::swap(v1, v2) – то же самое.
- Операторы ==, !=, <, >, <=, >= – лексикографическое сравнение.

Задание на лабораторную работу:

1. Описать класс обертку вокруг std::vector реализующий стек (LIFO).
2. Реализовать программу, моделирующую формирование поезда с двумя типами грузов - «Дерево», «Сталь»:
    a. Реализовать формирование первичного грузового поезда с двумя типами вагонов с консоли
    b. При помощи стекового класса промоделировать Т-образный сортировочный узел, разделяющий поезд на два, каждый поезд со своим типом груза
    c. Вывести содержимое поездов
3. Отобразить выделенные вами классы на диаграмме классов (без учета связей)