**ЛЕКЦИЯ 10**

**ТВОРЧЕСКИЙ КОНКУРС**

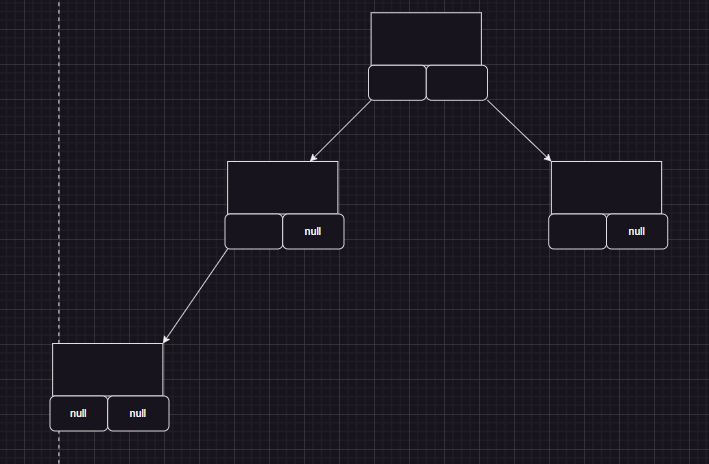
**СОЗДАНИЕ АВТОМАТИЗИРОВАННОГО РАБОЧЕГО МЕСТА**

1. Работаем до 15 мая!!
2. Язык C++, можно использовать windows forms и openGL

**БИНАРНЫЕ ДЕРЕВЬЯ**

1. Дерево – нелинейная структура, форма – связанные узлы;
2. Узел – единица хранения данных, имеющая указатели на следующие узлы;
3. Дочерний узел – узел, который находится непосредственно под узлом верхнего уровня по указателю;
4. Родительский узел – узел верхнего уровня, связан с потомками указателя;
5. Корневой узел не имеет родительских узлов;
6. Лист – узел, не имеющий потомков;
7. Высота дерева – число уровней до корня до листьев.

**СТРУКТУРА БИНАРНОГО ДЕРЕВА**



**МЕТОДЫ ПРОХОЖДЕНИЯ ДЕРЕВА**

1. Прямой обход:

- Обработать корень;

- Затем левое поддерево;

- Обойти правое поддерево.

1. Симметричный обход:

- Обработать лево

- Корень

- Право

1. Обратный обход:

- Лево

- Право

- Корень

**ИДЕАЛЬНО СБАЛАНСИРОВАННОЕ ДЕРЕВО**

1. Идеально сбалансированное дерево – когда для каждого узла левого и правого поддерева их кол-во отличается не более чем на 1;
2. ИБ дерево имеет минимальную высоту для n элементов. Вычисляется п формуле [log2n] + 1;

**БИНАРНОЕ ДЕРЕВО ПОИСКА**

1. Условие: значение левого потомка всегда меньше значения родителя, а правого потомка всегда больше родителя для каждого узла;

**ЛАБОРАТОРНАЯ РАБОТА**

1. Сделать UML диаграмму;
2. Осуществить все виды обходов;
3. Реализовать операции: вставка узла, удаление уза поиск элемент по ключу;
4. Визуализировать дерево с использованием любой графической библиотеки (opengl, sfml, sdl);
5. Пользовательский интерфейс разработать самостоятельно через windows forms;
6. В отчете дополнить содержание анализом задачи с разбором применения используемых структур данных и функций, в коде программы на C++ сделать подробные комментарии, скриншоты работы и визуализация решения.

**ЧТО Я ДОКЛАЖЫВАЮ?**

1. SFML

**ПОСЛЕДОВАТЕЛЬНЫЕ КОНТЕЙНЕРЫ STL БИБЛИОТЕКИ**

1. Делаем только 13;
2. STL – Standart Template Library. Состоит из набора контейнеров классов и набора стандартных алгоритмов;
3. Контейнер – объект, который содержит другие объекты одного типа;
4. Обобщенные алгоритмы (ОА) – обеспечение поиска, сортировки, слияния и т.д;
5. Контейнеры можно разделить на последовательные и ассоциативные.

**КОНТЕЙНЕРЫ**

1. Для использования контейнера необходимо включить необходимый заголовочный файл (#include <**название\_контейнера**>).
2. Базовые контейнеры: **векторы**, **списки**, **двусторонние очереди**. Спец. на основе базовых: **стэки**, **очереди**, **очереди с приоритетами**;
3. Ассоциативные контейнреры обеспечивают быстрый доступ к элементам по ключу: map (словарь), multimap (словарь с дубликатами), set (множество), multiset (множество с повторениями), bitset (бинарное множество);

**ВЕКТОР**

1. Аналог динамического ошибка;
2. Эффективно обрабатывает выработку элемента с помощью [];
3. Выделяется и удаляется автоматически;
4. Вставка элемента в любую позицию вектора, кроме конца, неэффективна;
5. Методы работы с памятью:

- size() – получить размер,

- capacity() – получить емкость,

- resize() – поменять размер вектора,

- reserve(int) – поменять емкость вектора,

- shrink\_to\_fit – поменять емкость вектора на её размер;

**DEQUE (ОЧЕРЕДЬ С ДВУМЯ КОНЦАМИ)**

1. Можно добавлять и удалять элементы как с начала, так и с конца;
2. Вывод может быть только через итераторы;

**LIST (СПИСОК)**

1. За базу берется двусвязный список;
2. Не поддерживает произвольный доступ через [];
3. Интересный конструктор: list(it t1, it t2) – создать список с элементами из диапазона t1 – t2 другого списка;
4. Для перемещения по списку использовать итераторы.

**FORWARD\_LIST (ОДНОНАПРАВЛЕННЫЙ СПИСОК)**

1. Встроенными методами можно получить только первый элемент;
2. max\_size() – получить кол-во элементов контейнера;
3. swap() – поменять значениями два списка.

**АДАПТЕРЫ КОНТЕЙНЕРОВ**

1. Специализированные последовательные контейнеры – стеки, очереди и очереди с приоритетом. Основаны на уже существующих контейнерах;

**СТЕКИ**

1. Работает по правилу “последний вошел – последний вышел”;
2. Можем обрабатывать только самый верхний элемент стека методом top;
3. Стек не дает использовать ни итераторы, ни [];
4. Может быть основан на deque, vector или list;
5. Образец: stack<**тип\_данных**, **тип\_контейнера**>;

**QUEUE**

1. “Первым вошел – первым вышел”;
2. Может быть основана на dequeue, векторе и списке;

**ОЧЕРЕДЬ С ПРИОРИТЕТАМИ**

1. Максимальный элемент контейнера всегда в начале;
2. По факту является отсортированным вариантом контейнера, на котором основана;
3. Может быть основана на векторе или списке;

**АССОЦИАТИВНЫЕ КОНТЕЙНЕРЫ STL**

1. Содержат пары **ключ:значение**;
2. Индексами выступают ключи, значения значениями;
3. Предоставляют доступ с помощью итераторов.

**MAP**

1. Контейнер пар ключ : значение;
2. Ключи отсортированы;
3. Не может быть два элемента с одинаковыми ключами;
4. Чтение:

**for (auto i : <словарь>) {}**

**СЛОВАРЬ С ДУБЛИКАТАМИ (MULTIMAP)**

1. Предусматривает хранение значений с одинаковыми ключами;

**SET (МНОЖЕСТОВ)**

1. Ассоциативный массив, в котором значения не имеют смысла;
2. Хранит элементы, упорядоченные по некоторому ключу;
3. Не может быть одинаковых элементов;
4. Элементы отсортированы;
5. Метод find() – возвращает итератор на найденное значение, иначе итератор конца множества;

**MULTISET (МНОЖЕТСВА С ДУБЛИКАТАМИ)**

1. Set, но с дубликатами. Все еще отсортирован;
2. Lower\_bound() –возвращает значение >= аргументу;
3. Upper\_Bound – аналогично;

**ФУЦНКЦИОНАЛЬНЫЕ ОБЪЕКТЫ**

1. Классы, в которых перегружен оператор ();

**ПРОДВИНУТЫЕ УКАЗАТЕЛИ**

1. Требуется подключить хедер <functional>;
2. Позволяют указывать как на объектные функции, так и на операторные;

**УКАЗАТЕЛИ НА МЕТОДЫ**

1. Пример: function<void(**название\_класса**)> **название\_указателя** = &**доступ\_к\_методу**;
2. Может конвертировать метод класса, в котором экземпляр класса задается неявно, в тот, где явно;

**АНОНИМНЫЕ ФУНКЦИОНАЛЬНЫЕ ОБЪЕКТЫ ИЛИ ЛЯМДА ВЫРАЖЕНИЯ**

1. Является удобной оберткой для функции.

**АДАПЕРЫ ФУНКЦИЙ**

1. Функция – которая принимает некоторую функцию и конструирует из неё другую;
2. Виды:

- Задает одному из параметров некоторое константное значение;

- Отрицатели для предикатов;

- Адаптеры указателей на функции;

- Адаптеры для методов класса;

**АЛГОРИТМЫ STL**

1. Каждый алгоритм является шаблонным и работает с любым видом контейнера;
2. Виды:

- Немод. операции с последователями;

- Модифицированные операции с последовательностями;

- Алгоритмы сортировки;

- Алгоритмы для работы с множествами и кучами (heap).

**ЛАБОРАТОРНАЯ РАБОТА 13**

1. Класс брать из лабораторной номер 3.