НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

Факультет < Программная инженерия >

Институт < ИИКС >

Лабораторная работа №3

Выполнил:

студент гр. Б17-514

Сапарбеков Султан

Преподаватель:

Трифоненков А.В.

Москва 2018

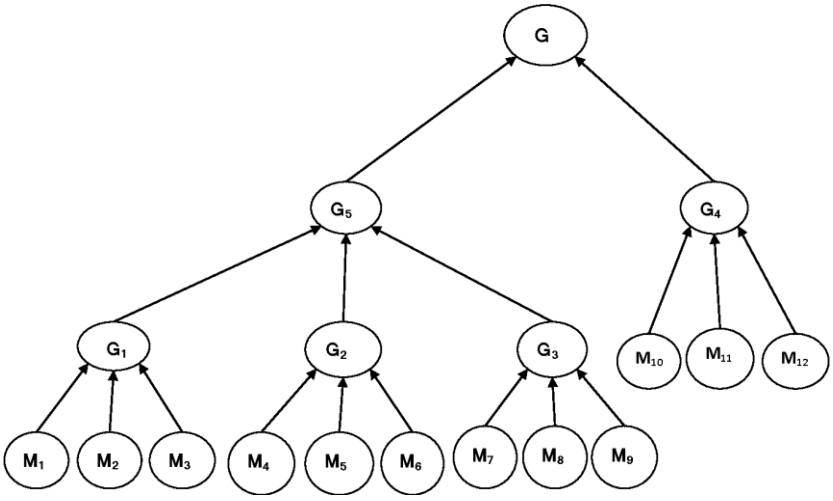
Вариант №18

Коллекция – 3-арное дерево

Типы хранимых данных – вещественные и комплексные числа, строки

Операции над коллекцией: map (построить новое дерево поэлементным преобразованием), where (построить новое дерево, в которое входят лишь те узлы исходного, которые удовлетворяют заданному условию), Слияние, Извлечение поддерева (по заданному элементу), Поиск на вхождение поддерева, Поиск элемента на вхождение.

3-арное дерево - это структура данных, в которой каждый узел имеет не более трех потомков (детей).



«Техническое задание»

-создать заголовочный файл и файл с реализациями: main.cpp и header.h;

-объявить структуру со следующими полями:

```
struct Node
{
        T data;
        Node<T>* left;
        Node<T>* right;
        Node<T>* middle;
};
```

«Пользовательский интерфейс»

1) Программа приглашает: 1) Выбрать тип данных, 2) Провести тест, 3) Узнать о программе, 4) Выйти из программы.
2) Нужно будет ввести количество элементов в дереве
3) Затем поэлементно заполняется дерево (первый элемент считается корнем дерева)
4) Программа предложит ввести операцию, которую хотите провести;

«Программный интерфейс»

Реализован тип данных Node. Поле data хранит элементы дерева, поля left, middle, right – указатели на соответствующие потомки дерева.

Реализованы классы: Complex и tree.

Класс Complex представляет концепцию: комплексных чисел. Поля double re, double im представляют собой действительные и мнимые части комплексного числа. Также в этом классе использованы перегрузка операторов.

Класс tree представляет 3-арное дерево. Поле root хранит указатель на корень дерева.

Полиморфизм реализован с помощью шаблонов.

«Тестирование»

Функции:

1) bool test_map();
2) bool test_where();
3) bool test_merger();
4) bool test_ejection_search_tree_and_search_elem();
5) bool test_map_c();
6) bool test_where_c();
7) bool test_merger_c();
8) bool test_ejection_search_tree_and_search_elem_c();
9) bool test_map_s();
10) bool test_merger_s();
11) bool test_ejection_search_tree_and_search_elem_s();
12) void test_general();

1-4 - тесты на соответствующие функции для вещественных чисел;

5-8 - тесты на соответствующие функции для комплексных чисел;

9-11 - тесты на соответствующие функции для строк;

Они все возвращают TRUE в случае удачного прохождения теста, FALSE в противном случае.

12 – функция, объединяющая все функции тестирования (1-11); Осведомляет об удачном или провальном прохождении теста.

Реализуемые функции и операции:

void freetree_el(); --- Удаляет дерево
void add_el(T element); --- Добавляет элемент к дереву
void print_el(); --- Выводит дерево
void map_quad_el(tree &); --- реализация функции map (возведение в квадрат) для вещественных и комплексных чисел
void map_str_el(tree &); --- реализация функции map (возведение в квадрат) для строк

void where_el(tree&); --- реализация функции where (создает новое дерево, элементы которого находятся в интервале от 5 до 50) для вещественных чисел.
void where_complex_el(tree &); --- реализация функции where (создает новое дерево, элементы которого находятся в интервале от re1 = 5 до re2 = 50) для комплексных чисел.
void merger_el(tree&); --- Слияние двух деревьев
void ejection_el(tree&, T); --- Извлечение поддерева по элементу
bool search_tree(tree &A); --- Поиск (введенного) дерева
bool search_el(T elem); --- Поиск элемента в дереве
bool equal_tree_el(tree&); --- Равенство деревьев (данная функция понадобилась для тестирования)