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

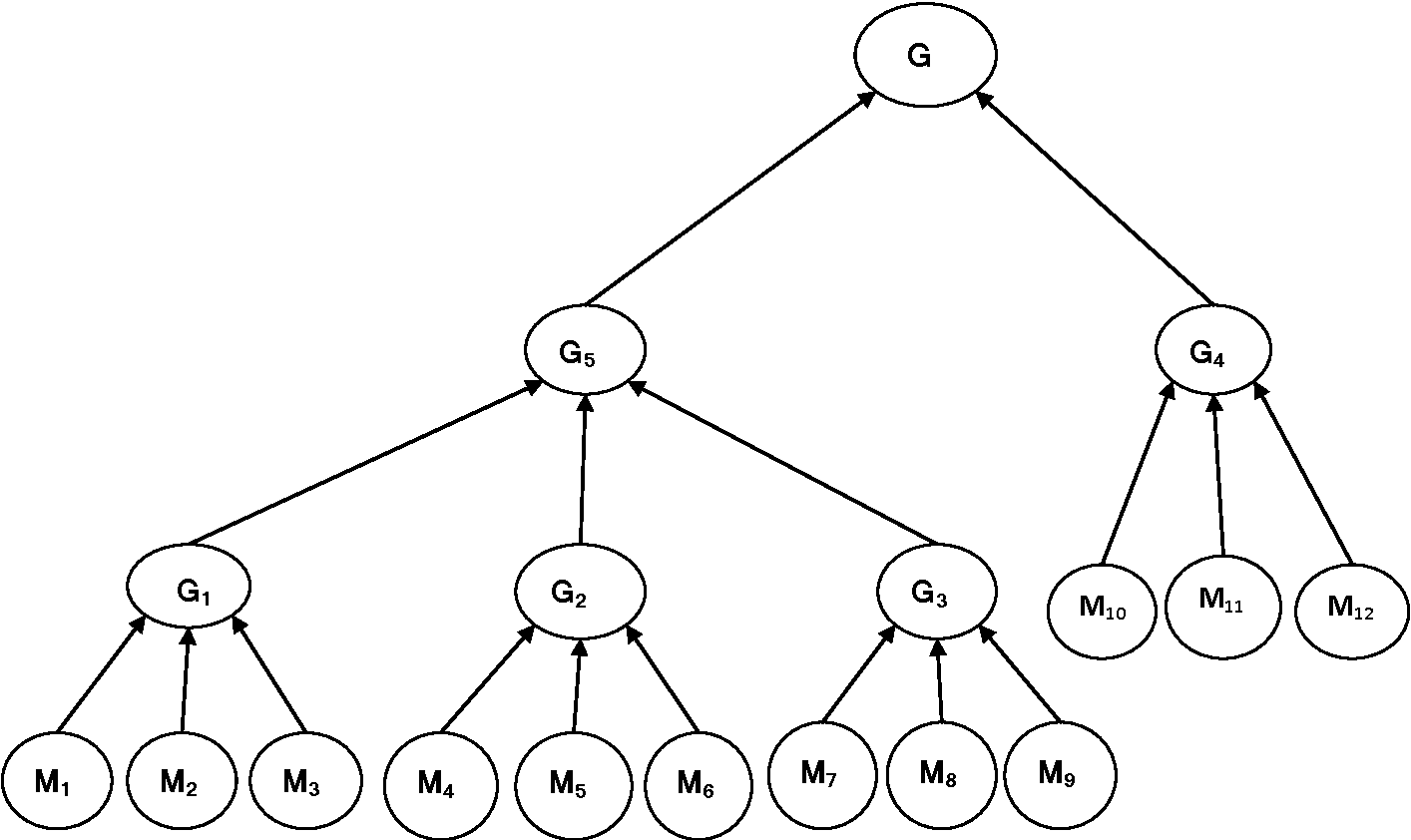
Вариант №18

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](https://ru.wikipedia.org/wiki/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: комплексныx 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)