

Лабораторная работа №4

по курсу информатики, 2 семестр

Варианты заданий

Постановка задачи

Написать на языке C++ реализацию абстрактного типа данных на основе структур данных типа дерево.

Минимальные требования к программе. В программе, в зависимости от варианта, требуется реализовать одну из следующих структур данных: бинарное дерево или 3-арное дерево. Для реализации необходимо использовать динамические структуры, основанные на указателях. Структура данных должна поддерживать работу с элементами различных типов (вообще говоря, произвольных, если они удовлетворяют некоторым условиям).

Для реализации необходимо использовать возможности ООП и шаблонов C++ (templates) – классов и функций. Во всех реализованных функциях необходимо обрабатывать случаи некорректных значений входных параметров – как правило, в таких случаях следует выбрасывать исключения.

Основные алгоритмы необходимо покрыть (модульными) тестами. Реализацию следует оснастить пользовательским интерфейсом (консольным) для проверки корректности реализации. **Выполнить тестирование скорости работы алгоритмов на больших (10^4 - 10^5 элементов) и очень больших (10^6 - 10^8) объемах данных.** Результаты оформить в виде графика зависимости времени выполнения от количества элементов.

Содержание вариантов

Типы-контейнеры:

№ варианта	Тип дерева	Типы хранимых элементов	Операции
1.	Бинарное дерево Обходы: –КЛП –КПЛ –ЛПК –ЛКП –ПЛК –ПКЛ К – корень Л – лево (левое поддерево) П – право (правое поддерево)	–Целые числа –Вещественные числа –Комплексные числа –Строки –Функции ²⁾ –Студенты ³⁾ –Преподаватели ³⁾	–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 –Сохранение в строку в
2.	n-арное дерево (n – фиксированные натуральное число) Различные варианты		

	обхода, например для $n=3$: K123 (т.е. корень, затем 1-е поддереву, потом 2-е и наконец 3-е).		соответствии с заданным обходом –Чтение из строки в соответствии с заданным обходом
3.	n -арное дерево (n – параметр, задаваемый на этапе конструирования дерева, основные обходы: КПЛ, КЛП, ПЛК, ЛПК*)	–	–Поиск узла по заданному пути, поиск по относительному пути
Производные типы данных:			
4.	Очередь с приоритетами	–Целые числа –Вещественные числа –Комплексные числа –Строки/символы –Функции –Студенты –Преподаватели	–map, where, reduce –Извлечение подпоследовательности (с i -го элемента по j -й) –Поиск на вхождение подпоследовательности –Слияние –Разделение (по заданному признаку) –Сохранение в строку и чтение из строки
5.	Множество		–map, where –объединение –пересечение –вычитание –проверка на включение подмножества –проверка на вхождение элемента –сравнение (равенство) двух множеств –Сохранение в строку и чтение из строки

¹⁾ Если $l = [a_1, \dots, a_n]$ – некоторый список элементов типа T , а $f: T \rightarrow T$, то:

$$\text{map}(f, l) \mapsto [f(a_1), \dots, f(a_n)]$$

Если, при тех же соглашениях, $h: T \rightarrow \text{Bool}$ – некоторая функция, возвращающая булево значение, то результатом $\text{where}(h, l)$ будет новый список l' , такой что: $a'_i \in l' \Leftrightarrow h(a'_i) = \text{true}$. Т.е. where фильтрует значения из списка l с помощью функции-фильтра h .

Функция reduce работает несколько иначе: «сворачивает» список в одно значение по заданному правилу $f: T \times T \rightarrow T$:

* Например, обход КЛП: сначала посещаем корень, а потом узлы слева направо.

$$\text{reduce}(f, l, c) \mapsto f \left(a_n, \left(f \left(a_{n-1}, \left(\dots f \left(a_2, (f(a_1 c)) \right) \right) \right) \right) \right)$$

где c – константа, «стартовое» значение. Например, $l = [1, 2, 3]$, $f(x_1, x_2) = 2x_1 + 3x_2$, тогда:

$$\begin{aligned} \text{reduce}(f, [1, 2, 3], 4) &= f(3, f(2, f(1, 4))) = \\ &= 2 \cdot 3 + 3(2 \cdot 2 + 3(2 \cdot 1 + 3 \cdot 4)) = \\ &= 2 \cdot 3 + 3(2 \cdot 2 + 3 \cdot 14) = 2 \cdot 3 + 3 \cdot 42 = 132 \end{aligned}$$

2) Точнее, указатели на функции. Ниже – минимальный пример, как создать «список функций»:

```
const int array_length = 3;
int(**f)(int) = malloc(array_length * sizeof(int(*) (int)));
f[0] = &inc1;
f[1] = &inc2;
f[2] = &inc3;
for (int index = 0; index < length; index++)
    printf("%i ", f[index](0));
// Вывод: 1 2 3
```

3) Точнее, описывающие их структуры. Персона характеризуется набором атрибутов, таких ФИО, дата рождения, некоторый идентификатор (в роли которого может выступать: номер в некотором списке, номер зачетки/табельный номер, номер паспорта, и др.). Пример структуры, описывающей персону:

```
class Person {
private:
    PersonID id;
    char* firstName;
    char* middleName;
    char* lastName;
    time_t birthDate;
public:
    PersonID GetID();
    char* GetFirstName();
    ...
}
```

Тип PersonID предназначен для идентификации персоны и может быть объявлен различным образом, в зависимости от выбранного способа идентификации человека. Если для этих целей используется, скажем, номер паспорта, можно предложить, по крайней мере, два различных определения:

первое:

```
#typedef Person_ID char* // null-terminated string† вида "0982 123243"
```

второе:

```
#typedef Person_ID struct { // можно и в виде класса
    int series;           // как вариант, char*
    int number;           // как вариант, char*
}
```

Для получения значения атрибутов предусматривают соответствующие методы, например:

```
char* name = person->getName(); // = "Иван"
```

```
char* fullName = oerson->getFullName(); // = "Иван Иванович Иванов",  
вычисляемый атрибут
```

5) Многочлен степени n записывается в виде: $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ и может быть однозначно задан списком своих коэффициентов a_0, \dots, a_n . Многочлен является функцией, на множестве функций определена ассоциативная операция – композиция \circ : $(f \circ g)(x) = f(g(x))$.

6) Перекодирование состоит в замене каждого символа на другой, получаемый с помощью функции кодирования, которая передается в качестве аргумента.

7) Подразумевается многочлен первой степени от n переменных: $F_n(x_1, \dots, x_n) = a_0 + a_1 x_1 + \dots + a_n x_n$.

Состав задачи формируется каждым студентом индивидуально путем выбора из перечисленных ниже пунктов.

№	Составляющие задачи	Рейтинг
1.	Бинарное дерево поиска	
	Базовые операции: вставка, поиск, удаление	5
1.1.	Балансировка	2
1.2.	map, reduce, where	3
1.3.	Прошивка	
1.3.1.	по фиксированному обходу	3
1.3.2.	по обходу, задаваемому параметром метода	4
1.4.	Сохранение в строку	
1.4.1.	по фиксированному обходу	1

[†] См. например: https://en.wikipedia.org/wiki/Null-terminated_string. Идея такая, что конец строки определяется по наличию символа с кодом 0.

1.4.2.	по обходу, задаваемому строкой форматирования (например: «{K}(Л)[П]»)	3
1.5.	Чтение из строки	
1.5.1.	по фиксированному обходу	1
1.5.2.	по обходу, задаваемому строкой форматирования (например: «{K}(Л)[П]»)	4
1.5.3.	в формате списка пар «узел-родитель»	3
1.6.	Извлечение поддерева (по заданному корню)	2
1.7.	Поиск на вхождение поддерева	3
2.	Бинарная куча	
	Базовые операции: вставка, поиск, удаление	
2.1.	Варианты реализации:	
2.1.1.	через указатели на узлы	5
2.1.2.	через массив	5
2.2.	Извлечение поддерева (по заданному элементу)	2
2.3.	Поиск на вхождение поддерева	3
2.4.	Сохранение в строку	
2.4.1.	по фиксированному обходу	1
2.4.2.	по обходу, задаваемому строкой форматирования (например: «{K}(Л)[П]»)	3
2.4.3.	в формате списка пар «узел-родитель»	2
2.5.	Чтение из строки	
2.5.1.	по фиксированному обходу	1
2.5.2.	по обходу, задаваемому строкой форматирования (например: «{K}(Л)[П]»)	4
2.5.3.	в формате списка пар «узел-родитель»	3
3.	3-арное дерево	
	Базовые операции: вставка, поиск, удаление	6

3.1.	map, reduce	3
3.2.	Извлечение поддерева (по заданному элементу)	2
3.3.	Поиск на вхождение поддерева	3
3.4.	Сохранение в строку	
3.4.1.	по фиксированному обходу	1
3.4.2.	по обходу, задаваемому строкой форматирования (например: «{K}(1)[2]{3}»)	3
3.4.3.	в формате списка пар «узел-родитель»	3
3.5.	Чтение из строки	
3.5.1.	по фиксированному обходу	2
3.5.2.	по обходу, задаваемому строкой форматирования (например: «{K}(1)[2]{3}»)	4
3.5.3.	в формате списка пар «узел-родитель»	3
3.6.	Поиск узла по заданному полному (абсолютному) пути, поиск по относительному пути	2
3.7.	Реализация дерева поиска	3
4.	n-арное дерево	
	Базовые операции: вставка, поиск, удаление	7
4.1.	map, reduce	1
4.2.	Извлечение поддерева (по заданному элементу)	2
4.3.	Поиск на вхождение поддерева	3
4.4.	Сохранение в строку	
4.4.1.	по фиксированному обходу	1
4.4.2.	по обходу, задаваемому строкой форматирования (например: «{K}(1)[2]{3}»)	3
4.4.3.	в формате списка пар «узел-родитель»	2
4.5.	Чтение из строки	
4.5.1.	по фиксированному обходу	2

4.5.2.	по обходу, задаваемому строкой форматирования (например: «{K}(1)[2]{3}»)	3
4.5.3.	в формате списка пар «узел-родитель»	3
4.6.	Поиск узла по заданному полному (абсолютному) пути, поиск по относительному пути	1
4.7.	Реализация дерева поиска	4
5.	Очередь с приоритетами	
	Базовые операции: вставка, поиск, удаление	
5.1.	Варианты реализации:	
5.1.1.	на базе бинарной кучи	2
5.1.2.	на базе бинарного дерева поиска	2
5.2.	map, reduce, where	2
5.3.	Извлечение подпоследовательности (с i-го элемента по j-й)	1
5.4.	Поиск на вхождение подпоследовательности	2
6.	Множество	
	Базовые операции: вставка, поиск, удаление	
6.1.	Варианты реализации:	
6.1.1.	на базе бинарной кучи	2
6.1.2.	на базе бинарного дерева поиска	2
6.1.3.	на базе 3-арного дерева	2
6.1.4.	на базе n-арного дерева	2
6.2.	map, reduce, where	2
6.3.	Операции над множествами: объединение, пересечение, вычитание	2
6.4.	Проверка на включение (подмножества), на равенство (двух множеств)	2
6.5.	Сохранение в строку и чтение из строки	3

7.	Общее	
	Реализация общих интерфейсов (см. ЛР-2)	
7.1.	ICollection	2
7.2.	IEnumerable, реализация TreeEnumerator	3
7.3.	Перегрузка операторов	2
8.	Прикладные задачи	
	Реализация общих интерфейсов (см. ЛР-2)	
8.1.	ICollection	2
8.2.	IEnumerable, реализация TreeEnumerator	3
8.3.	Перегрузка операторов	2

Минимальная необходимая сумма рейтинга – 25, с учетом прохождения онлайн-курса – 12. Превышение этой величины дает бонус к итоговой оценке за работу.

Пояснения

¹⁾ Например, «{1}K[2][3]» означает обход 1K23, при этом элементы первого дерева должны быть ограничены символами { слева и } справа, второго – символами [и], и третьего – символами [и].

Критерии оценки

1.	Качество программного кода:	<ul style="list-style-type: none"> – стиль (в т.ч.: имена, отступы и проч.) (0-2) – структурированность (напр. декомпозиция сложных функций на более простые) (0-2) – качество основных и второстепенных алгоритмов (напр. обработка граничных случаев и некорректных исходных данных и т.п.) (0-3) 	0-6 баллов
2.	Качество пользовательского интерфейса:	<ul style="list-style-type: none"> – предоставляемые им возможности (0-2) – наличие ручного/автоматического ввода исходных данных (0-2) – настройка параметров для автоматического режима – отображение исходных данных и промежуточных и конечных результатов и др. (0-2) 	0-6 баллов

3.	Качество тестов	<ul style="list-style-type: none"> – степень покрытия – читаемость – качество проверки (граничные и некорректные значения, и др.) – полнота и качество представления результатов тестирования 	0-5 баллов
4.	Полнота выполнения задания и качество ТЗ	Оценивается качество подготовки ТЗ, функциональная полнота реализации,	0-3 баллов
5.	Владение теорией	знание алгоритмов, области их применимости, умение сравнивать с аналогами, оценить сложность, корректность реализации	0-5 баллов
6.	Оригинальность реализации	оцениваются отличительные особенности конкретной реализации – например, общность структур данных, наличие продвинутых графических средств, средств ввода-вывода, интеграции с внешними системами и др.	0-5 баллов
		Итого	0-30 баллов
7.	Объем выбранного задания	дополнительная работа, выполненная сверх установленного минимума, согласованность выбранных составляющих (например, нескольких взаимосвязанных задач оценивается выше, чем реализация набора независимых задач)	

Для получения зачета за выполнения лабораторной работы необходимо соблюдение всех перечисленных условий:

- оценка за п. 1 должна быть не менее 3 баллов
- оценка за п. 4 должна быть не менее 3 баллов
- оценка за п. 5 должна быть больше 0
- суммарная оценка за работу без учета п. 6 должна быть не менее 17 баллов