Фонд оценочных средств  
по дисциплине  
«Информатика (информатика и программирование, архитектура вычислительных систем, проектирование человеко-машинного интерфейса)»

Приложение 3

Лабораторная работа №3

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 ПОИСКА или 3-арное дерево ПОИСКА. 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.

Содержание вариантов

Типы-контейнеры:

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Тип дерева** | **Типы хранимых элементов** | **Операции** |
|  | 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 * Функции2) * Студенты3) * Преподаватели3) | * 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 соответствии с заданным обходом * Чтение из строки в соответствии с заданным обходом * Поиск узла по заданному пути, поиск по относительному пути |
|  | n-арное дерево  (n – фиксированные натуральное число)  Различные варианты обхода, например для n=3: К123 (т.е. корень, затем 1-е поддерево, потом 2-е и наконец 3-е). |
| Производные типы данных: | | | |
|  | 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 |
|  | 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 – некоторый список элементов типа , а , то:

Если, при тех же соглашениях, – некоторая функция, возвращающая булево значение, то результатом будет новый список , такой что: . Т.е. where фильтрует значения из списка с помощью функции-фильтра .

Функция reduce работает несколько иначе: «сворачивает» список в одно значение по заданному правилу :

где – константа, «стартовое» значение. Например, , , тогда:

**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[[1]](#footnote-1) 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”, вычислимый атрибут

**5)** Многочлен степени записывается в вде: и может быть однозначно задан списком своих коэффициентов . Многочлен является функцией, на множестве функций определена ассоциативная операция – композиция : .

**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 переменных: .

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Тип коллекции** | **Типы хранимых элементов** | **Примечания** |
| *Группа А* | | | |
|  | Бинарное дерево | * Целые числа * Вещественные числа * Строки | Обходы: КЛП, ЛПК |
|  | Бинарное дерево | * Строки * Функции | Обходы: КПЛ, ЛКП |
|  | Бинарное дерево | * Вещественные числа * Комплексные числа * Строки | Обходы: ЛПК, ПЛК |
|  | Бинарное дерево | * Студенты * Преподаватели | Обходы: ЛКП, ПКЛ |
|  | Бинарное дерево | * Функции * Студенты * Преподаватели | Обходы: ПЛК, КЛП |
|  | Бинарное дерево | * Целые числа * Вещественные числа * Комплексные числа | Обходы: ПКЛ, КПЛ |
|  | Бинарное дерево | * Студенты * Преподаватели | Обходы: КЛП, ЛПК |
|  | Бинарное дерево | * Вещественные числа * Комплексные числа | Обходы: КПЛ, ЛКП |
|  | Бинарное дерево | * Вещественные числа * Комплексные числа | Обходы: ЛПК, ПЛК |
|  | Бинарное дерево | * Комплексные числа * Строки * Функции | Обходы: ЛКП, ПКЛ |
|  | Бинарное дерево | * Целые числа * Вещественные числа * Строки | Обходы: ПЛК, КЛП |
| *Группа Б* | | | |
|  | 3-арное дерево | * Целые числа * Вещественные числа * Комплексные числа | Обходы1):  (К)[2]{1}{3} |
|  | 3-арное дерево | * Функции * Студенты * Преподаватели | Обходы1):  К(3)[2][1] |
|  | 3-арное дерево | * Целые числа * Комплексные числа * Строки | Обходы1):  К(1)(3)(2) |
|  | 3-арное дерево | * Функции * Студенты * Преподаватели | Обходы1):  [1][2]{3}К |
|  | 3-арное дерево | * Вещественные числа * Комплексные числа * Строки | Обходы1):  {2}{1}{3}{К} |
|  | 3-арное дерево | * Строки * Функции | Обходы1):  |3||2||1|К |
|  | 3-арное дерево | * Вещественные числа * Комплексные числа * Строки | Обходы1):  (1)К[2][3] |
|  | 3-арное дерево | * Функции * Студенты * Преподаватели | Обходы1):  [1][2]К(3) |
|  | 3-арное дерево | * Целые числа * Комплексные числа | Обходы1):  {3}К{2}[1] |
|  | 3-арное дерево | * Целые числа * Комплексные числа | Обходы1):  [3][2]К(1) |
|  | 3-арное дерево | * Студенты * Преподаватели | Обходы1):  [2]К(1)[3] |
| Задания повышенной сложности | | | |
|  | Словарь слов | * Символы * Строки | Построение «именного указателя»: для заданной строки определить список входящих в нее слов (возможно, за исключением некоторых, перечисленных в заданном словаре), и для каждого такого найденного слова указать список позиций в исходной строке, в которых оно встречается. Для поиска элементов по ключу использовать бинарное дерево (ключом будет слово, а позиция в строке – значением). |
|  | 3-арное дерево  Обход задается строкой1) | * Целые числа * Строки | Операции – все, кроме where и поиска на вхождение поддерева |
|  | Множество  (для реализации использовать бинарное дерево) | * Целые числа * Студенты * Преподаватели | Все операции |
|  | Множество  (для реализации использовать 3-арное дерево) | * Целые числа * Вещественные числа * Строки | Все операции |
|  | Очередь с приоритетами  (для реализации использовать бинарное дерево) | * Целые числа * Студенты * Преподаватели | Чтение и запись в строку можно реализовать только для случая, когда элементы - числа |
|  | Очередь с приоритетами  (для реализации использовать бинарное дерево) | * Комплексные числа * Студенты * Преподаватели | Чтение и запись в строку можно реализовать только для случая, когда элементы - числа |
|  |  |  |  |

Пояснения

1) Например, «{1]К[2][3}» означает обход 1К23, при этом элементы первого дерева должны быть ограничены символами { слева и ] справа, второго – символами [ и ], и третьего – символами [ и }.

Критерии оценки

|  |  |  |  |
| --- | --- | --- | --- |
|  | Качество программного кода: | * 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) | 0-5  баллов |
|  | Качество пользовательского интерфейса: | * 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) | 0-5  баллов |
|  | Качество тестов | * степень покрытия * читаемость * качество проверки (граничные и некорректные значения, и др.) | 0-5  баллов |
|  | Полнота выполнения задания и качество ТЗ | Оценивается качество подготовки ТЗ, полнота выполнений минимальных требований | 0-5  баллов |
|  | Владение теорией | знание алгоритмов, области их применимости, умение сравнивать с аналогами, оценить сложность, корректность реализации | 0-3  баллов |
|  | 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 |

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

1. См. например: https://en.wikipedia.org/wiki/Null-terminated\_string. Идея такая, что конец строки определяется по наличию символа с кодом 0. [↑](#footnote-ref-1)