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

Приложение 2

Лабораторная работа №2

по курсу информатики, 2 семестр

Варианты заданий

Постановка задачи

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

**Минимальные требования к программе**. В программе, в зависимости от варианта, требуется реализовать одну из следующих структур данных: линейный связанный список (односвязный или двусвязный, линейный или кольцевой), очередь, стек или дек. Для реализации необходимо использовать концепцию связанных списков. 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. Для операций, таких как сцепление списков, следует предусмотреть перегрузку операторов. 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

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 1. Список типов-контейнеров** | | | |
| **№ варианта** | **Тип коллекции** | **Типы хранимых элементов** | **Операции** |
|  | Линейный список | * Целые числа * Вещественные числа * Комплексные числа * Строки * Функции2) * Студенты3) * Преподаватели3) | * Сортировка * map, where, reduce1) * Конкатенация * Извлечение подпоследовательности (по заданным индексам) * Поиск на вхождение подпоследовательности * Слияние * Разделение (по заданному признаку) |
|  | Линейный односвязный |
|  | Линейный двусвязный |
|  | Кольцевой односвязный |
|  | Кольцевой двусвязный |
|  | Очередь | * map, where, reduce * Конкатенация * Извлечение подпоследовательности (по заданным индексам) * Поиск на вхождение подпоследовательности * Слияние * Разделение (по заданному признаку) |
|  | Стек |
|  | Дек | * Сортировка * map, where, reduce * Конкатенация * Извлечение подпоследовательности (по заданным индексам) * Поиск на вхождение подпоследовательности * Слияние * Разделение (по заданному признаку) |
| Производные типы данных: | | | |
|  | Многочлен4) | Коэффициенты:   * Целые числа * Вещественные числа * Комплексные числа | Сложение, умножение, умножение на скаляр, вычисление значения для заданного значения аргумента, композиция |
|  | «Линейная форма»5) | Коэффициенты:   * Целые числа * Вещественные числа * Комплексные числа | Сложение (и вычитание), умножение на скаляр, вычисление значения при заданных значениях аргументов |
|  | Очередь с приоритетами | * Целые числа * Вещественные числа * Комплексные числа * Строки/символы * Функции * Студенты * Преподаватели | * map, where, reduce * Конкатенация * Извлечение подпоследовательности (по заданным индексам) * Поиск на вхождение подпоследовательности * Слияние * Разделение (по заданному признаку) |
|  | Поток (stream) данных | * map, where, reduce * Извлечение подпоследовательности (по заданным индексам) * Поиск на вхождение подпоследовательности * Слияние * Разделение (по заданному признаку) |
|  | Множество | * map, where * объединение * пересечение * вычитание * проверка на включение подмножества * проверка на вхождение элемента * сравнение (равенство) двух множеств |

**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: .

|  |  |  |  |
| --- | --- | --- | --- |
| **Таблица 2. Содержание вариантов заданий** | | | |
| **№ варианта** | **Тип коллекции** | **Типы хранимых элементов** | **Примечания\*)** |
|  | Линейный односвязный | * Целые числа * Вещественные числа * Строки |  |
|  | Линейный двусвязный | * Строки * Функции |  |
|  | Кольцевой односвязный | * Вещественные числа * Комплексные числа * Строки |  |
|  | Кольцевой двусвязный | * Студенты * Преподаватели |  |
|  | Очередь | * Функции * Студенты * Преподаватели |  |
|  | Стек | * Целые числа * Вещественные числа * Комплексные числа |  |
|  | Дек | * Студенты * Преподаватели |  |
|  | Многочлен | * Вещественные числа * Комплексные числа |  |
|  | «Линейная форма» | * Вещественные числа * Комплексные числа |  |
|  | Очередь с приоритетами | * Комплексные числа * Строки * Функции |  |
|  | Поток (stream) данных | * Целые числа * Вещественные числа * Строки |  |
|  | Множество | * Целые числа * Вещественные числа * Комплексные числа |  |
|  | Линейный односвязный | * Функции * Студенты * Преподаватели |  |
|  | Линейный двусвязный | * Целые числа * Комплексные числа * Строки |  |
|  | Кольцевой односвязный | * Функции * Студенты * Преподаватели | * map, reduce * Конкатенация * Извлечение подпоследовательности * Слияние |
|  | Кольцевой двусвязный | * Вещественные числа * Комплексные числа * Строки |  |
|  | Очередь | * Строки * Функции |  |
|  | Стек | * Вещественные числа * Комплексные числа * Строки |  |
|  | Дек | * Функции * Студенты * Преподаватели |  |
|  | Многочлен | * Целые числа * Комплексные числа |  |
|  | «Линейная форма» | * Целые числа * Комплексные числа |  |
|  | Очередь с приоритетами | * Студенты * Преподаватели |  |
|  | Поток (stream) данных | * Студенты * Преподаватели |  |
|  | Множество | * Студенты * Преподаватели |  |
|  | Линейный односвязный | * Целые числа * Функции * Студенты |  |
|  | Линейный двусвязный | * Функции * Студенты * Преподаватели |  |
|  | Кольцевой односвязный | * Комплексные числа * Студенты * Преподаватели |  |
|  | Кольцевой двусвязный | * Строки * Студенты * Преподаватели |  |
|  | Очередь | * Функции * Студенты * Преподаватели |  |
|  | Стек | * Студенты * Преподаватели |  |
| Задания повышенной сложности | | | |
|  | Словарь слов (список) | * 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. |
|  | «Ханойская башня» | Написать программу, решающую задачу о «ханойской башне»[[2]](#footnote-2). Стержни моделировать стеками, в роли колец могут выступать произвольные предметы, характеризующиеся формой и цветом. Параметрами задачи являются список предметов и номер стержня, на котором из предметы размещены изначально. | |
|  | Интерпретатор регулярных выражений | Исходными являются две строки, представленные список символов. Первая строка – регулярное выражение[[3]](#footnote-3). Определить, соответствует ли ему вторая строка. | |
|  | «Сортировочная станция» | На станцию пришел поезд, составленный из пронумерованных (например, по возрастанию) вагонов различных типов. Требуется пересобрать состав таким образом, чтобы вагоны одного типа шли строго последовательно и с сохранением исходной нумерации. Вагоны можно отцеплять от состава только с переднего края, а прицеплять – только в конец. Станция оснащена несколькими тупиковыми путями. Решить задачи с использованием возможно меньшего количества тупиков. Пример. Пусть – типы вагонов; каждый вагон представлен парой «тип-номер» . Тогда «исходный» состав из 8-ми вагонов может быть представлен списком: . В результате работы алгоритма должен получиться состав: . | |
|  | Строка | Реализовать класс строк CString, включая различные операции: конкатенацию, получение подстроки, поиск вхождений подстроки, разбиение на строки, замена подстроки на другую строку, и др. С помощью наследования и переопределения методов, представить два варианта реализации: на основе связанных списков и на основе динамических массивов. | |
|  |  |  |  |

\*) Если ничего не указано, то реализуется весь список операций, указанных в таблице 1.

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) | 0-7  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) | 0-6  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 | 0-5  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 | | 0-33  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)
2. См. https://ru.wikipedia.org/wiki/ханойская\_башня или https://habrahabr.ru/post/200758 [↑](#footnote-ref-2)
3. https://ru.wikipedia.org/wiki/регулярное\_выражение [↑](#footnote-ref-3)