

Лабораторная работа №1

по курсу информатики, 2 семестр

Варианты заданий

Постановка задачи

Написать на языке C реализацию абстрактного типа данных – полиморфной коллекции на основе динамического массива. Написать программу-оболочку для тестирования этой реализации. Под полиморфным, в данном случае, подразумевается такой массив, который может хранить (и обрабатывать) значения различных типов. Что конкретно понимается под обработкой – зависит от конкретного варианта задания.

Минимальные требования к программе. В программе, в зависимости от варианта, требуется реализовать одну из следующих структур данных: вектор, квадратная матрица, прямоугольная матрица, список (или один из производных от них). Для реализации необходимо использовать концепцию динамических массивов. Структура данных должна поддерживать работу с элементами различных типов: вообще говоря, произвольных, если они удовлетворяют некоторым условиям, которые зависят от специфики задачи. Т.е. система должна быть «открытой»: должно быть можно использовать любой тип, для которого предоставлены реализации необходимых операций.

Основные алгоритмы необходимо покрыть (модульными) тестами. Реализацию следует оснастить пользовательским интерфейсом (консольным) для проверки корректности реализации.

Содержание вариантов

Типы-контейнеры:

№ варианта	Тип коллекции	Примеры типов хранимых элементов	Операции
1.	Вектор	–Целые числа –Вещественные числа –Комплексные числа	–Векторное сложение, скалярное произведение
2.	Квадратная матрица	–Целые числа –Вещественные числа –Комплексные числа	–Матричное сложение и умножение, умножение на скаляр, прибавление к строке линейной комбинации других строк ¹⁾
3.	Прямоугольная матрица	–Целые числа –Вещественные числа –Комплексные числа	–Матричное сложение и умножение, транспонирование, прибавление к строке линейной комбинации других строк
4.	Динамический массив	–Целые числа –Вещественные числа	–Сортировка –map, where, reduce ²⁾

		–Комплексные числа –Строки –Функции ³⁾ –Студенты ⁴⁾ –Преподаватели ⁴⁾	–Конкатенация
Производные типы данных:			
5.	Многочлен ⁵⁾	Коэффициенты: –Целые числа –Вещественные числа –Комплексные числа	Сложение, умножение, умножение на скаляр, вычисление значения для заданного значения аргумента, композиция
6.	Строка	Символы	Конкатенация, получение подстроки (с i-го символа по j-й), поиск подстроки (сравнение может быть чувствительным к регистру, так и нет), перекодирование строки ⁶⁾
7.	«Линейная форма» ⁶⁾	Коэффициенты: –Целые числа –Вещественные числа –Комплексные числа	Сложение (и вычитание), умножение на скаляр, вычисление значения при заданных значениях аргументов

¹⁾ Т.е. функцию, которая в качестве аргумента принимает номер i строки, к которой требуется прибавить линейную комбинацию, и список коэффициентов β_1, \dots, β_m для остальных строк. Если матрица $A = (a_{ij})$ имеет размерность $m \times n$, то результатом действия функции будет новая матрица, у которой i -я строка имеет вид: $\bar{a}_i + \sum_k \beta_k \bar{a}_k$. Пример сигнатуры функции:

`Matrix* AddLinearCombination(Matrix* matrix, int rowIndex, void* alphas)`

Тип `void*` у списка коэффициентов `alphas` обусловлен тем, что, в общем случае, эти коэффициенты могут быть любого числового типа (целое, вещественное, комплексное – в зависимости от того, что указано в варианте задания).

²⁾ 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 $f: T \times T \rightarrow T$:

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

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

$$\text{reduce}(f, [1, 2, 3], 4) = f(3, f(2, f(1, 4))) =$$

$$= 2 \cdot 3 + 3(2 \cdot 2 + 3(2 \cdot 1 + 3 \cdot 4)) =$$

$$= 2 \cdot 3 + 3(2 \cdot 2 + 3 \cdot 15) = 2 \cdot 3 + 3 \cdot 49 = 153$$

3) 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
```

4) Точнее, описывающие их структуры. Персона характеризуется набором атрибутов, таких как 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:

```
struct Person {
    Person_ID id;
    char* firstName;
    char* middleName;
    char* lastName;
    time_t birthDate;
}
```

Тип `Person_ID` 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 string1 вида "0982 123243"
```

второе:

```
#typedef Person_ID struct {
    int series;        // как вариант, char*
    int number;        // как вариант, char*
}
```

Для получения значения атрибутов предусматривают соответствующие функции, например:

```
char* name = getName(person); // = "Иван"
char* fullName = getFullName(person); // = "Иван Иванович Иванов",
вычисляемый атрибут
```

¹ См. например: https://en.wikipedia.org/wiki/Null-terminated_string. Идея такая, что конец строки определяется по наличию символа с кодом 0.

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: $F_n(x_1, \dots, x_n) = a_0 + a_1 x_1 + \dots + a_n x_n$.

№ варианта	Тип коллекции	Типы хранимых элементов	Операции
1.	Вектор	–Целые числа –Вещественные числа	Векторное сложение, скалярное произведение
2.	Динамический массив	–Целые числа –Вещественные числа	–Сортировка –map, where –Конкатенация
3.	Квадратная матрица	–Целые числа –Вещественные числа	Матричное сложение и умножение, умножение на скаляр, прибавление к строке линейной комбинации других строк
4.	Прямоугольная матрица	–Целые числа –Вещественные числа	Матричное сложение и умножение, транспонирование, прибавление к строке линейной комбинации других строк
5.	Динамический массив	–Целые числа –Комплексные числа	–map, where –Конкатенация
6.	Многочлен	Коэффициенты –Целые числа –Вещественные числа	Сложение, умножение, умножение на скаляр, вычисление значения для заданного значения аргумента, композиция
7.	Строка	Символы	Конкатенация, получение подстроки (с i -го символа по j -й), поиск подстроки (реализовать два варианта: чувствительное к регистру сравнение, и нечувствительное)
8.	Динамический массив	–Вещественные числа –Комплексные числа	–Сортировка –map, where –Конкатенация
9.	«Линейная форма»	Коэффициенты: –Целые числа	Сложение (и вычитание), умножение на скаляр,

		–Вещественные числа	вычисление значения при заданных значениях аргументов
10.	3-мерный вектор	–Вещественные числа –Комплексные числа	Векторное сложение, скалярное произведение, векторное произведение
11.	Динамический массив	–Целые числа –Строки	–Сортировка –map, where –Конкатенация
12.	Квадратная матрица	–Вещественные числа –Комплексные числа	Матричное сложение и умножение, умножение на скаляр
13.	Прямоугольная матрица	–Вещественные числа –Комплексные числа	Матричное сложение и умножение, транспонирование
14.	Динамический массив	–Вещественные числа –Функции	–map, where –Конкатенация
15.	Многочлен	Коэффициенты: –Целые числа –Комплексные числа	Сложение, умножение, умножение на скаляр, вычисление значения для заданного значения аргумента
16.	Строка	Символы	Конкатенация, получение подстроки (с i-го символа по j-й), перекодирование строки
17.	Динамический массив	–Студенты –Преподаватели	–map, where –Конкатенация
18.	«Линейная форма»	Коэффициенты: –Вещественные числа –Комплексные числа	Сложение (и вычитание), умножение на скаляр, вычисление значения при заданных значениях аргументов
19.	Вектор	–Вещественные числа –Комплексные числа	Векторное сложение, скалярное произведение
20.	Динамический массив	–Строки –Функции	–map, where –Конкатенация
21.	Квадратная матрица	–Целые числа –Комплексные числа	Матричное сложение и умножение, умножение на скаляр
22.	Прямоугольная матрица	–Целые числа –Комплексные числа	Матричное сложение и умножение, транспонирование
23.	Динамический массив	–Строки –Функции	–map, where, reduce –Конкатенация
24.	Многочлен	Коэффициенты: –Вещественные числа –Комплексные числа	Сложение, умножение, умножение на скаляр, вычисление значения для заданного значения аргумента

			аргумента
25.	Строка	–Символы –Строки	Конкатенация, получение элементов с i-го по j-й (подмассив), разбиение строки на слова
26.	Динамический массив	–Вещественные числа –Строки	–Сортировка –map, where –Конкатенация
27.	«Линейная форма»	Коэффициенты: –Вещественные числа –Комплексные числа	Сложение (и вычитание), умножение на скаляр, вычисление значения при заданных значениях аргументов
28.	Вектор	–Целые числа –Комплексные числа	Векторное сложение, скалярное произведение
29.	Строка	Символы: строка содержит запись какого-либо из следующих чисел: целого вещественного или комплексного	Конкатенация, получение подстроки (с i-го символа по j-й), преобразование в/из числа.
30.	Квадратная матрица (трехленточная ²)	–Вещественные числа –Комплексные числа	Матричное сложение и умножение, умножение на скаляр
Задания повышенной сложности			
31.	Динамический массив, строка	–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.
32.			

Критерии оценки

1.	Качество программного кода:	<ul style="list-style-type: none"> – стиль (в т.ч.: имена, отступы и проч.) (0-3) – структурированность (напр. декомпозиция сложных функций) 	0-8 баллов
----	-----------------------------	--	------------

² Т.е. от нуля отличны только элементы, стоящие на главной диагонали, а также на диагоналях непосредственно над и под ней.

		на более простые) (0-3) – качество основных и второстепенных алгоритмов (напр. обработка граничных случаев и некорректных исходных данных и т.п.) (0-3)	
2.	Качество тестов	– степень покрытия – читаемость – качество проверки (граничные и некорректные значения, и др.)	0-8 баллов
3.	Полнота выполнения задания	Оценивается в целом соответствие предложенной реализации требованиям, самостоятельность выполнения задания.	0-5 баллов
4.	Владение теорией	– понимание алгоритмов работы с динамическими массивами – понимание арифметики указателей – владение методикой работы с void*	0-4 баллов
5.	Обработка ошибок	– Простота идентификации ошибки – Простота локализации ошибки – Простота и полнота расшифровки содержания/причины ошибки	0-5 баллов
		Итого	0-30 баллов
<i>Бонусные задачи</i>			
6.	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
7.	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

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

- основные структуры данных и алгоритмы должны быть реализованы
- для основных алгоритмов должны быть написаны тесты

-
- суммарная оценка за работу без учета пп. 5-6 должна быть не менее 18 баллов