**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: “Двусвязные списки, динамические массивы”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

Изучение внутреннего строения двусвязного списка и строения динамического массива, понимания, как они хранятся в памяти.

**Основные теоретические положения.**

*Динамические массивы*

Для того чтобы создать в динамической области некоторый объект, необходима одна обычная переменная-указатель (не динамическая переменная). Сколько таких объектов понадобится для одновременной обработки, столько необходимо иметь обычных переменных-указателей. Таким образом, проблема задач неопределенной размерности созданием одиночных динамических объектов решена быть не может.

Решить эту проблему поможет возможность создавать в динамической области памяти массивы объектов с таким количеством элементов, которое необходимо в данный момент работы программы, т. е. создание динамических массивов. Действительно, для представления массива требуется всего одна переменная-указатель, а в самом массиве, на который ссылается этот указатель, может быть столько элементов, сколько требуется в данный момент времени.

Для создания одномерного динамического массива используется следующий синтаксис инструкции new (стиль С++):

int \*Arr = new int [100];

Причем в этом случае оператор new также возвращает указатель на объект типа int - первый элемент в созданном массиве.

Освободить динамическую область от этого массива можно с помощью инструкции delete:

delete [] Arr;

После этого занятый участок памяти будет возвращен в список свободной памяти и может быть повторно использован для размещения других динамических объектов.

*Двусвязные списки*

Одномерный однонаправленный список представляет собой совокупность отдельных элементов, каждый из которых содержит две части – информационную (Data) и адресную (Tail).

Информационная часть предназначена для хранения полезных данных и может иметь практически любой тип. Адресная часть каждого элемента содержит адрес следующего элемента списка.

Для работы со списком достаточно знать только адрес его первого элемента (Beg). Зная адрес первого элемента списка можно последовательно получить доступ к любому другому его элементу.

Поскольку каждый элемент списка должен иметь две части, логичнее всего представить его в виде следующей структуры:

struct list {

int data;

list \*tail;

};

Типовыми операциями при работе со списками являются:

1) создание списка;

2) освобождение памяти от списка (удаление списка);

3) доступ к заданному элементу списка для манипуляций с его информационной частью;

4) добавление нового элемента к списку;

5) удаление элемента из списка;

6) перестановка элемента списка на новую позицию внутри списка.

Достоинством подобных структур является простота добавления, удаления и перестановки элементов списка, которые осуществляются путем манипуляций с адресными частями без перезаписи всего списка.

*Сравнение*

Сравним производительность различных операций для списков с массивами. Основное преимущество массивов – возможность обратиться к произвольному элементу по его индексу. Для списка подобная операция потребует просмотра всех предшествующих элементов. Однако добавление в начало и середину массива, а также удаление из начала и середины требует сдвига всех элементов массива, расположенных после вставляемого (удаляемого). Для списка же производительность операций добавления и удаления не зависит ни от длины списка, ни от места вставки. Сортировка вставками в списке имеет такую же производительность, что и простые виды сортировок для массивов (например, пузырьковая сортировка или сортировка выбором).

Таким образом, если в задаче требуется структура с частыми операциями вставки и удаления в середине, следует предпочесть список, если же важнее быстрый доступ по индексу – следует выбрать массив. При выборе структуры следует также учитывать, что для организации односвязного списка на каждый элемент требуется одно поле связи, занимающее 4 байта (для массива подобные накладные расходы отсутствуют). С другой стороны, список может динамически расширяться во время работы программы, в то время как под массив отводится фиксированное количество элементов, которое необходимо указать на этапе компиляции.

**Постановка задачи.**

Необходимо реализовать программу, которая выполняет следующие действия.

1. Формирование двусвязного списка размерности N, где:

a) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы списка, N определяется автоматически по количеству введенных элементов;

2. Определение скорости создания двусвязного списка п. 2.

3. Вставка, удаление, обмена и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.

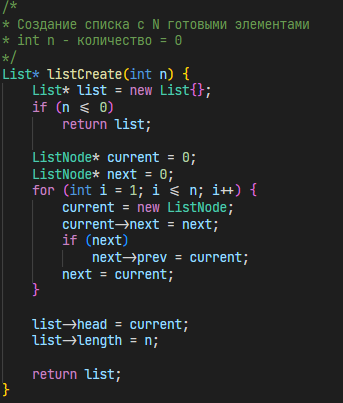
4. Определение скорости вставки, удаление и получения элемента двусвязного списка п. 3.

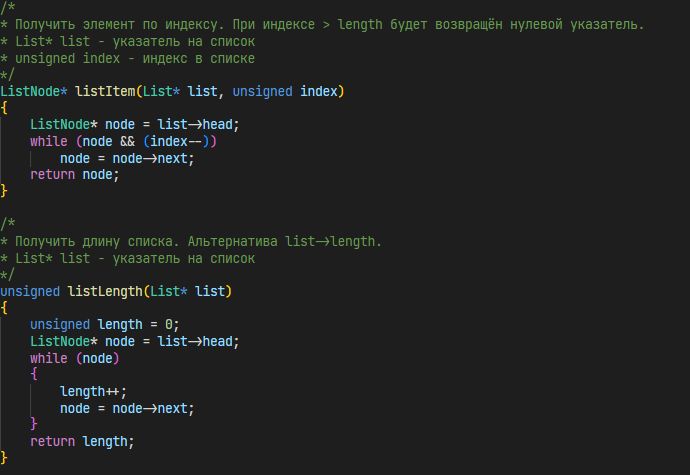
Должна быть возможность запуска каждого пункта многократно, если есть возможность (если в списке/массиве нет элементов, то нельзя ничего удалить и об этом нужно сообщить пользователю). Необходимо сравнить результаты. Для этого пункты 1–4 должны принимать одинаковые значения.

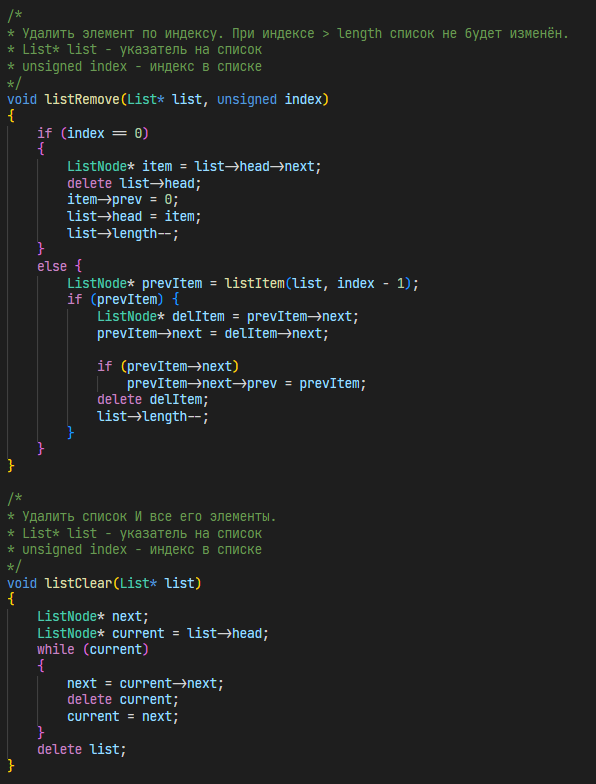
**Выполнение работы.**

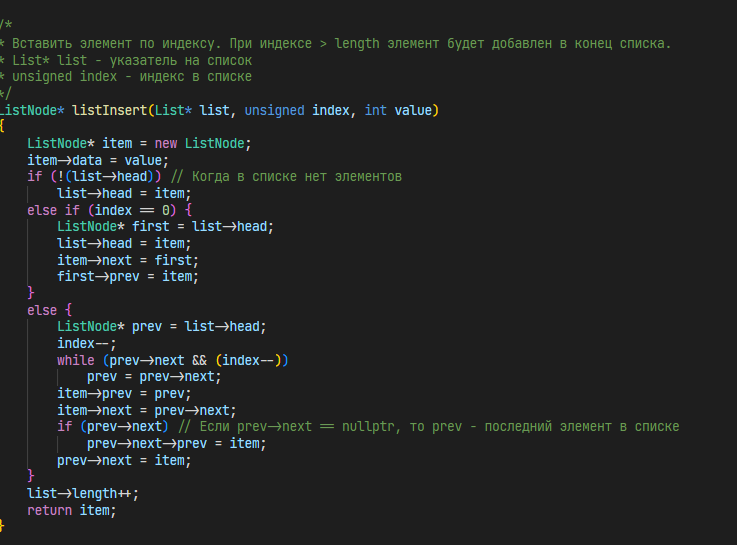
Структура списка, функции списка:

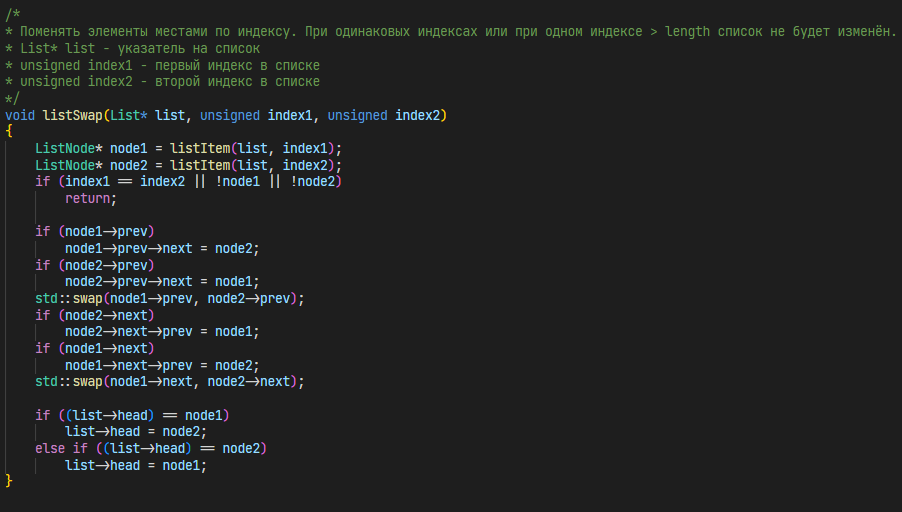




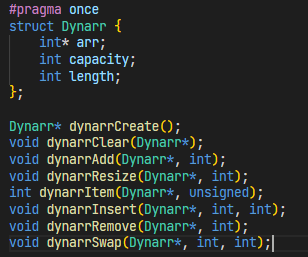


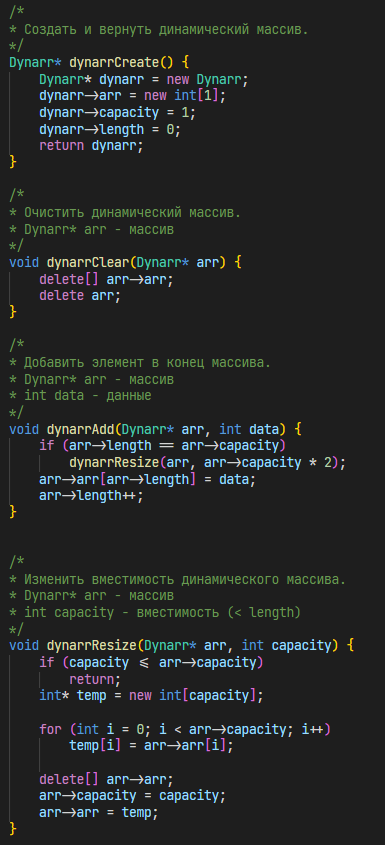


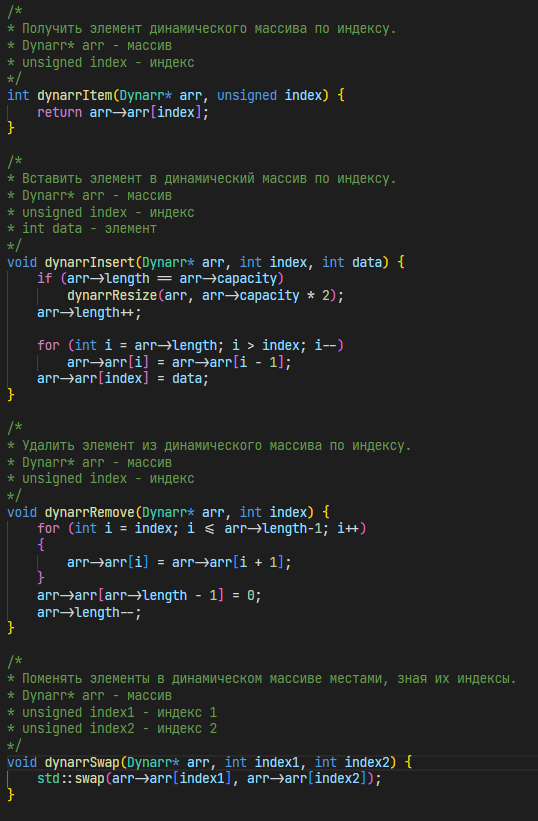




Структура динамического массива и его функции:



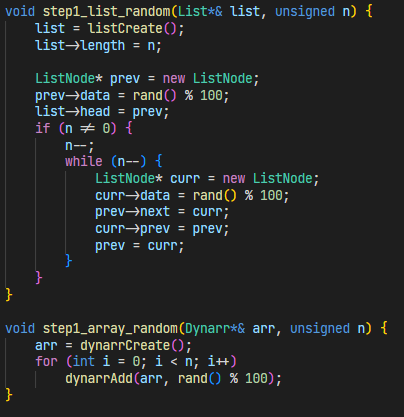




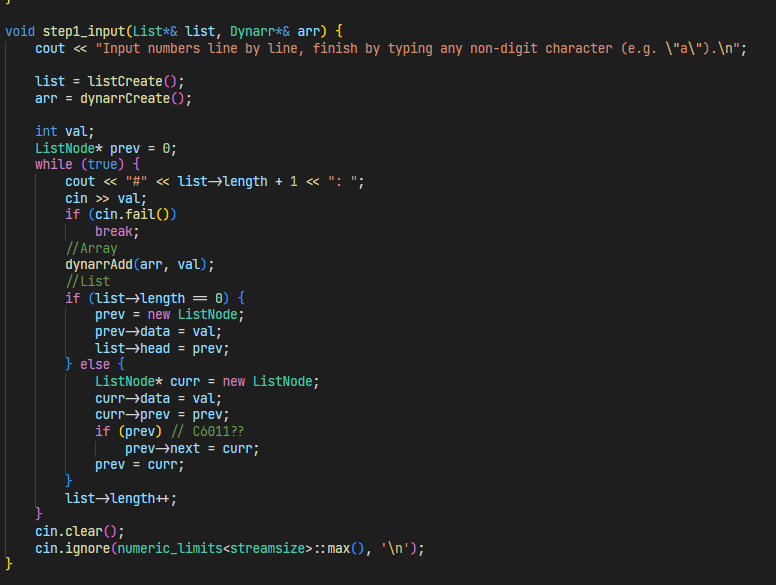
Вспомогательные функции ввода и вывода:



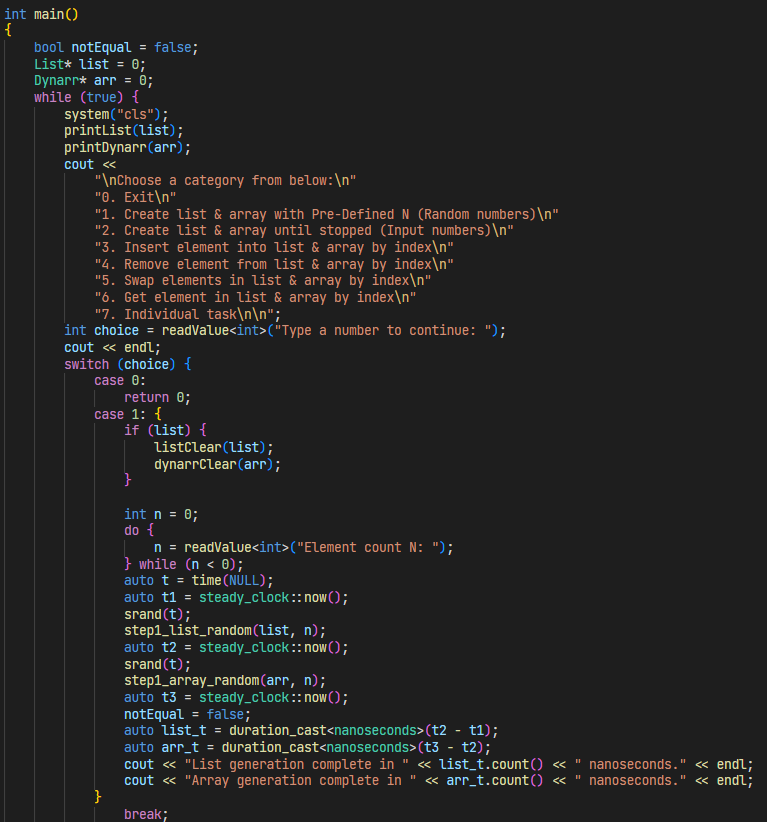
Задача 2, ввод случайных чисел для обоих типов структур

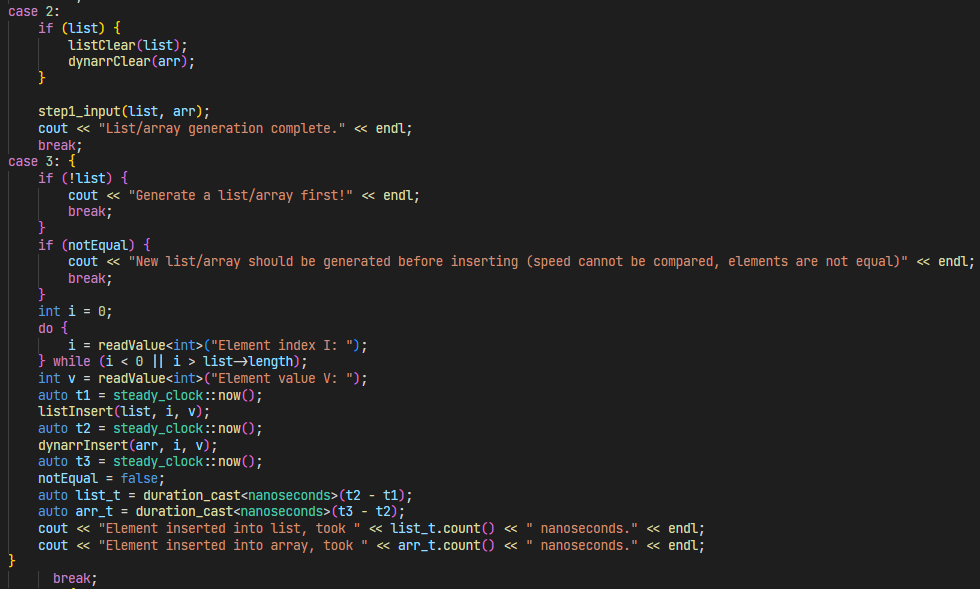


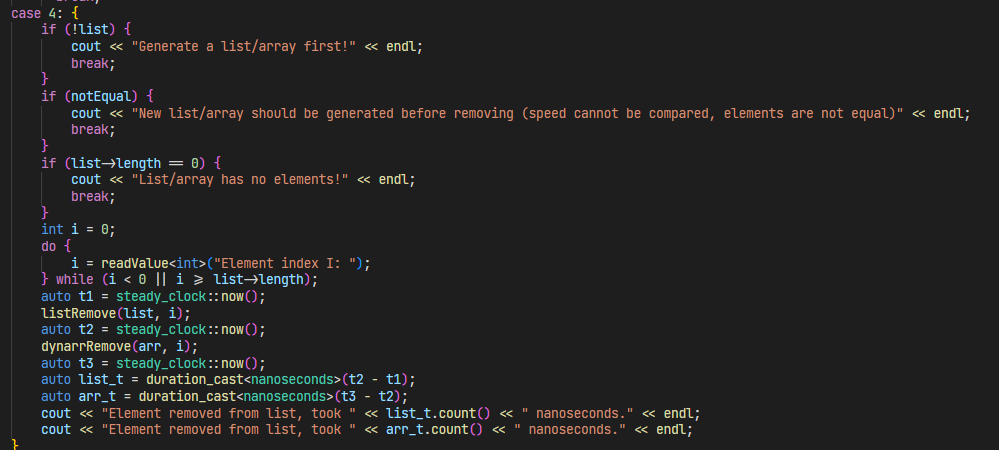
Задача 2, ввод с клавиатуры

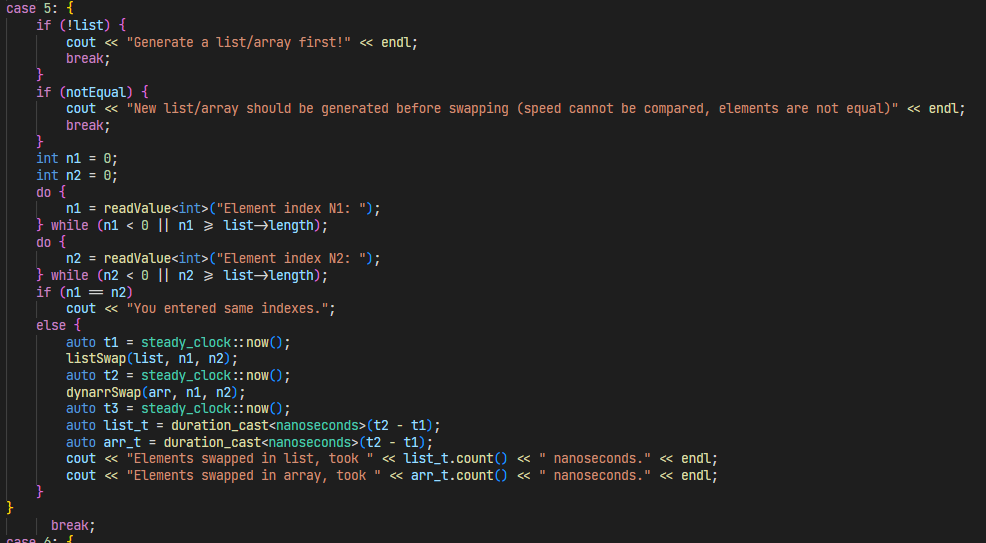


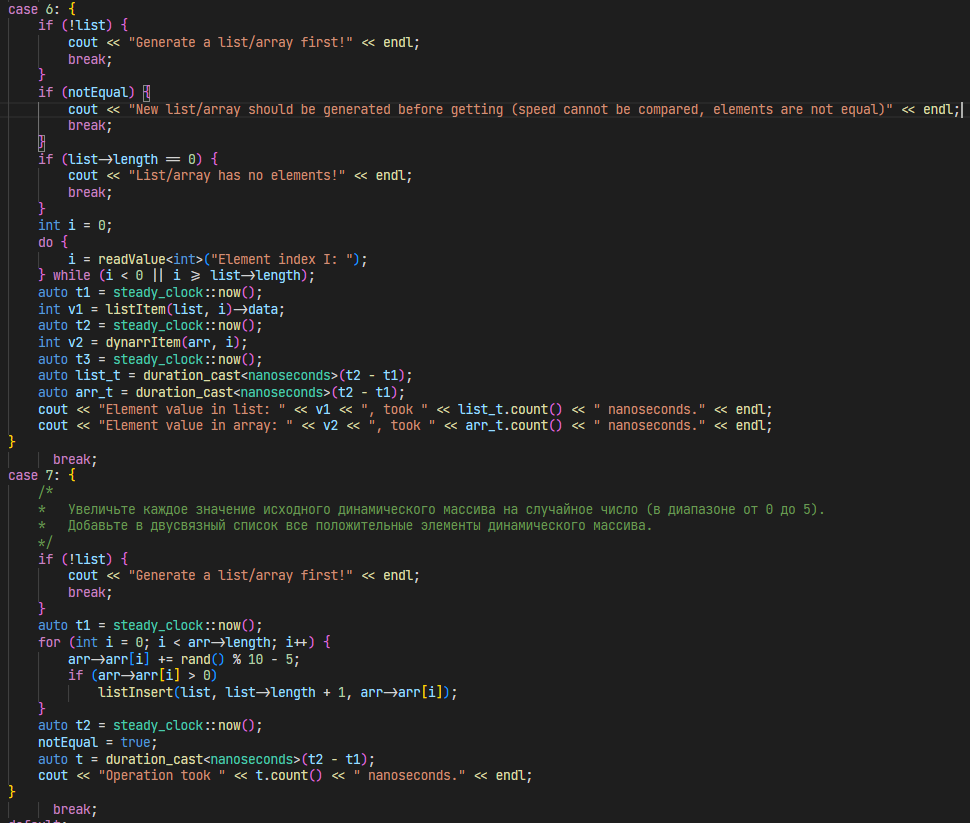
Главная функция:

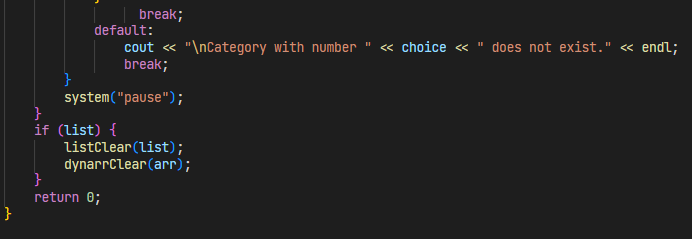












**Выводы.**

Я научился работать со списками и динамическими массивами, изучил их строение и нашёл плюсы и минусы их использования в разных ситуациях.