Лабораторная работа №4

«Безопасная работа с динамической памятью в C++»

Задание 1.

Реализовать простейший аналог std::shared_ptr и std::unique_ptr.

Класс shared_ptr обязан иметь управляющий блок, который будет считать количество объектов shared_ptr так же ссылающихся на данный объект. **Объект не может быть удалён пока счётчик ссылок больше 1**. С принципами работы указателя можно ознакомиться https://learn.microsoft.com/ru-ru/cpp/standard-library/shared-ptr-class?view=msvc-170.

Класс unique_ptr является единственным владельцем объекта. При передаче прав на объект **обязательно используется семантика перемещения** C++. С принципами работы указателя можно ознакомиться https://learn.microsoft.com/ru-ru/cpp/standard-library/unique-ptr-class?view=msvc-170.

Классы обязаны быть шаблонными.

Задание 2.

Реализовать динамическую библиотеку String (аналог std::string) с использованием простейшего итератора и собственных умных указателей (т.к. библиотека динамическая, то указатели в данном задании не являются шаблонными).

Реализовать и продемонстрировать с помощью визуальных компонентов следующие функции:

- `void* memcpy(void* s1, const void* s2, size_t n);`
- `void* memmove(void* s1, const void* s2, size_t n);`
- `char* strcpy(char* s1, const char* s2);`
- `char* strncpy(char* s1, const char* s2, size_t n);`
- `char* strcat(char* s1, const char* s2);`
- `char* strncat(char* s1, const char* s2, size_t n);`
- `int memcmp(const void* s1, const void* s2, size_t n);`
- `int strcmp(const char* s1, const char* s2);`
- `int strcoll(const char* s1, const char* s2);`
- `int strncmp(const char* s1, const char* s2, size_t n);`
- `size_t strxfrm(char* s1, const char* s2, size_t n);`
- `har* strtok(char* s1, const char* s2);`
- `void* memset(void* s, int c, size_t n);`
- `char* strerror(int errnum);`
- `size_t strlen(const char* s);`

*не забывайте про реализацию нужных конструкторов и операторов копирования (Правило трех).

Задание 3.

Написать парсер C++ кода на языке C++.

Парсер должен выдать следующую информацию:

1. Количество переменных каждого типа и их названия. Если у переменных есть базовое значение - вывести. Базовые параметры функции не учитывать.
2. Указать сколько классов, структур, массивов было инициализировано в коде.
3. Выдать на экран список прототипов функций (для функции, не имеющей прототипа, также должен быть выведен прототип)
4. Выдать координату (номер строки, индекс строки) каждого изменения любой переменной, в т.ч. массива.
5. Подсчитать количество локальных переменных и выдать их координаты.
6. Подсчитать количество перегруженных функций и выдать их координаты.
7. Рассчитать глубину каждого ветвления (отсчёт с 1).
8. Вывести на экран логические ошибки, которые не зависят от действий во время выполнения программы. Пример: const bool a = true; while (a){} или while (false){}.

Ввод данных сделать двумя вариантами:

1. Открыть с помощью диалогового окна файл .cpp или .h
2. В текстовый блок оконного приложения вставить код.

Форматирование кода должно быть по стандарту google и предусматривать 2 варианта объявления указателей: int* p и int *p;

https://google.github.io/styleguide/cppguide.html;

Темы для защиты лабораторной работы: Умные указатели в C++, семантика копирования и перемещения в C++, использование шаблонов в динамических библиотеках, константные указатели, регулярные выражения.