Лабораторная работа №4

«Итераторы. Библиотека String и Vector»

Задание 1.

Реализовать struct pair. Предусмотреть вариант pair<pair<T, T>, pair<T, T>> a;
Работа задания должна быть продемонстрирована в задании 3.

Задание 2.

Реализовать динамическую библиотеку String. Решить нижеприведённую задачу из первой лабораторной с помощью визуальных компонентов. 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).

Задача для задания 2:

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:

9. Открыть файл .CPP
10. В текстовый блок QT 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;

Задание 3.

Решить нижеприведённую задачу из 1 лабораторной работы с помощью визуальных компонентов на самописном Vector не используя стандартные библиотеки. Реализовать динамическую библиотеку Vector (на шаблонах) и итератор для Vector. В библиотеке vector необходимо реализовать и продемонстрировать работу следующих функций с помощью визуальных компонентов:

- assign; Удаляет вектор и копирует указанные элементы в пустой вектор.
- at; Возвращает ссылку на элемент в заданном положении в векторе.
- back; Возвращает ссылку на последний элемент вектора.
- begin; Возвращает итератор произвольного доступа, указывающий на первый элемент в векторе.
- capacity; Возвращает число элементов, которое вектор может содержать без выделения дополнительного пространства.
- cbegin; Возвращает постоянный итератор произвольного доступа, указывающий на первый элемент в векторе.
- cend; Возвращает константный итератор произвольного доступа, указывающий на позицию, следующую за концом вектора.
- crbegin; Возвращает константный итератор, который указывает на первый элемент в обратном векторе.
- crend; Возвращает константный итератор, который указывает на последний элемент в обратном векторе.
- clear; Очищает элементы вектора.
- data; Возвращает указатель на первый элемент в векторе.
- emplace; Вставляет элемент, созданный на месте, в указанное положение в векторе.

- emplace_back; Добавляет элемент, созданный на месте, в конец вектора.
- empty; Проверяет, пуст ли контейнер вектора.
- end; Возвращает итератор произвольного доступа, который указывает на конец вектора.
- erase; Удаляет элемент или диапазон элементов в векторе из заданных позиций.
- front; Возвращает ссылку на первый элемент в векторе.
- insert; Вставляет элемент или множество элементов в заданную позиции в вектор.
- max_size; Возвращает максимальную длину вектора.
- pop_back; Удаляет элемент в конце вектора.
- push_back; Добавляет элемент в конец вектора.
- rbegin; Возвращает итератор, указывающий на первый элемент в обратном векторе.
- rend; Возвращает итератор, который указывает на последний элемент в обратном векторе.
- reserve; Резервирует минимальную длину хранилища для объекта вектора.
- resize; Определяет новый размер вектора.
- size; Возвращает количество элементов в векторе.
- swap; Меняет местами элементы двух векторов.

Задача для задания 3: Брюс недавно получил работу в NEERC (Numeric Expression Engineering & Research Center), где изучают и строят много различных любопытных чисел. Его первым заданием стало исследование двудесятичных чисел.

Натуральное число называется **двудесятичным**, если его десятичное представление является суффиксом его двоичного представления; и двоичное и десятичное представление рассматривается без ведущих нулей. Например, 1010 = 10102, так что **10** двудесятичное число. Числа 101010 = 11111100102 и 4210 = 1010102 не являются двудесятичными.

Сначала Брюс хочет создать список двудесятичных чисел. Помогите ему найти **n**-ое наименьшее двудесятичное число.

Входные данные

Одно целое число **n** (**1 ≤ n ≤ 10 000**).

Выходные данные

Вывести одно число - **n**-ое наименьшее двудесятичное число в десятичном представлении.

Входные данные #1	**Выходные данные #1**
1	1
Входные данные #2	**Выходные данные #2**
2	10
Входные данные #3	**Выходные данные #3**
10	1100

Лимит времени 1 секунда