1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт Кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 4**

1. **«Подсчёт количества шестнадцатеричных чисел в программе»**
2. по дисциплине «Структуры данных»
3. Выполнил
4. студент гр. 4851003/10002 Билан Н.С.
6. *<подпись>*
7. Преподаватель Семьянов П. В.

*<подпись>*

**Цель работы**

Разработать программу, подсчитывающую количество различных идентификаторов в программе на языке С/С++, используя для этого самую подходящую структуру данных.

**Задачи**

* Выбрать подходящую структуру данных
* Реализовать её на языке С/C++
* Использовать эту реализацию в нужной программе
* Проверить созданную программу

**Введение**

Для того, чтобы подсчитать количество идентификаторов, необходимо создать ассоциативный массив, ключом в котором будет сам идентификатор, а значением – количество встреч в коде. Реализовать ассоциативный массив (словарь) можно используя различные структуры данных, поэтому необходимо выбрать наиболее эффективную конкретно для этой задачи.

**Подход к решению**

Программа должна один раз «проходить» по анализируемой программе и при нахождении идентификатора добавлять его в словарь. Можно сделать предположение, что если число встретилось недавно, то вероятность его повторного появления будет больше, чем у тех, которые встречались раньше, ведь они могли быть использованы в работе цикла. Таким образом, чтобы добавлять повторяющиеся идентификаторы в словарь максимально быстро, необходимо, чтобы они были максимально близки к началу структуры. Иными словами, наиболее вероятные идентификаторы должны быть максимально близки к корню. Это требование лучше всего удовлетворяет самобалансирующееся дерево (splay-дерево), оно и было реализовано в программе.

**Реализация решения**

1. Создание структуры *node* дерева, которая содержит ключ (сам идентификатор), значение (частоту) и ссылки на левого и правого потомка. Это дерево является деревом поиска, и поэтому ключ левого потомка будет меньше ключа родителя, а ключ правого потомка будет больше ключа родителя.
2. Реализация вспомогательных операций левого и правого поворота (операции LL\_ROTATE и RR\_ROTATE).
3. Реализация главной операции Splay(). Она находит с ключом , а затем делает её новым корнем, используя операции поворота LL\_ROTATE и RR\_ROTATE. Если же с таким ключом в дереве не было, то splay() возвращала последнее значение, к которому был получен доступ (иными словами, потенциального родителя ).
4. Реализация операции вставки в дерево. Функция сначала делает операцию Splay() с данным ключом, затем, если ключ нового корня совпадает со значением полученного числа, то частота этого корня увеличивается на один. Если же ключ нового корня не совпадает со значением полученного числа, то это означает, что такое число встретилось впервые, поэтому оно добавляется в дерево, и его частота становится равна 1.
5. Реализация функции поиска. По сути это есть операция Splay().
6. Реализация функции «подсчёта» чисел. Программа посимвольно считывает значения из заданной на вход программы. Если она находит начало идентификатора, которое находится не в комментарии или строковом литерале, она записывает это число в буфер , который является строкой. По окончании этого идентификатора она добавляет его в дерево.
7. Реализация главной функции , которая считывает имя файла из второго аргумента и открывает этот файл, вызывает функцию подсчёта, добавляет ноды в объект класса дерева, а затем закрывает файл. Далее даёт возможность пользователю удалить или найти одну конкретную ноду.
8. Проведены тесты этой утилиты на разных файлах.

**Результат**

Написана и проверена утилита, которая подсчитывает количество различных идентификаторов, используя для этого splay-дерево, являющееся наиболее подходящей структурой данных для этой задачи.

**Вывод**

Программа работает относительно быстро, алгоритмическая сложность прохода по файлу (не учитывая вставки в дерево) - . Сложность вставки же, в свою очередь, , при этом чем больше повторяющихся чисел будет в программе, тем быстрее они будут добавляться в словарь. Такое свойство обеспечивает splay-дерево, выбранное в качестве наиболее подходящей структуры данных.