**Казахстанский филиал МГУ г. Нур-Султан**

КФ МГУ

Факультет Вычислительной Математики и Кибернетики

Отчет по второй семестровой работе

Семестровая работа с условиями (9,14,1)

От студента Айтеев Альнур Тулендыулы ВМК-11

Преподавателю Нетесову В.В.

Г. Нур-Султан

2019

Содержание

* 1. Введение
     1. Постановка задачи
     2. Варианты задания
     3. Требования к программе
  2. Теоретическая часть
     1. Программное обеспечение
     2. Section Data
     3. Section Bss
     4. Section Text
     5. Алгоритм работы
  3. Заключение
  4. Список использованной литературы
  5. Приложения

Введение

1. **ПОСТАНОВКА ЗАДАЧИ**

Дана последовательность от 1 до 20 слов, каждое из которых содержит от 1 до 8 заглавных латинских букв; соседние слова разделены запятой, за последним словом следует точка.

Требуется ввести эту последовательность и преобразовать ее во внутреннее представление, а затем напечатать по алфавиту определенные слова с дополнительной информацией о каждом из них.

**2. ВАРИАНТЫ ЗАДАНИЯ**

**А. Внутреннее представление последовательности слов:**

9) Двоичное дерево поиска *(в нем слева от каждой вершины-слова должны находиться только те слова, что предшествуют этому слову по алфавиту, а справа - следующие за ним по алфавиту).*

**Б. Какие слова и в каком порядке печатать:**

14)Все различные слова, входящие в последовательность более одного раза.

**Замечание:** в каждой группе слова печатать по алфавиту.

**В. Дополнительная информация о слове:**

1) Нет дополнительной информации.

**3. ТРЕБОВАНИЯ К ПРОГРАММЕ**

**1.** Для размещения звеньев списков (вершин дерева) выделить в памяти область подходящего размера — «кучу». Описать процедуру (аналогичную процедуре New языка Паскаль), которая при каждом обращении к ней выделяет из кучи свободные ячейки под новое звено (вершину).

**2.** Описать в программе следующие процедуры:

⎯ чтение очередного слова, дополнение его справа пробелами (до 8 символов) и запись его в фиксированное место памяти;

⎯ просмотр списка (дерева) и печать нужных слов.

**3.** Программа должна быть протестирована на различных исходных данных.

Теоретическая часть

1. **Программное обеспечение**

Для выполнения данной семестровой работы использовался стандартный компилятор gcc, включающий в себя достаточное количество пакетов и библиотек для работы на языке Си и языке ассемблера соответственно.

**Для запуска программы требуется (как минимум):**

- х64 Linux 16.04 и выше.

- 3 Кб свободного места в оперативной памяти

- 1 МБ свободного места на жестком диске

- Одноядерный процессор, достаточно мощный для поддержания работы операционной системы.

**Для компиляции программы требуется:**

- 3 Кб свободного места на жестком диске

- Версия gcc: 4.2.1. (По усмотрению)

- Версия текстового редактора GNU nano: 2.9.3. (По усмотрению)

1. **SECTION DATA**

В данном разделе описаны константы, применяемые в программе.

**СТРОКИ С ИНФОРМАЦИЕЙ**

fmt: “%s” – предназначен для ввода строк (слов).

noFreeSpace: “НЕТ СВОБОДНОЙ ПАМЯТИ” – строка.

pr1: “%s (” – элемент дерева.

pr2: “,” – разделение между левым и правым поддеревом.

pr3: “)” – закрытая скобка для листа дерева.

pr4: “\_\_\_ (” – слово, которое повторилось в предложении.

**СТРУКТУРА ДЕРЕВА**

.set word 0 – Начало слова

.set num 16 – Начало номера

.set left 24 – Начало левого указателя

.set right 32 – Начало правого указателя

Для структуры из-за выравнивания выделяется 48 байт памяти.

1. **SECTION BSS**

str 1000 – Вводимая строка

buffer 20 – Буфер для временного хранения слова

heap 1000 – Куча

heap\_ptr 8 – Указатель на текущую свободную память

tree\_ptr 8 – Указатель на корень дерева

tmp 8 – временная копия tree\_ptr

1. **SECTION TEXT**

**Замечание:** Каждая строчка кода прокомментирована. Ниже будет описано только предназначение меток и «не очевидных» строк кода.

**Метки:**

main: - базовая метка

Весь код до L1 предназначен для обьявления стека, создания кучи, ввода строки.

L1: - Начало анализа строки на наличие слов

Помещаем в %esi адрес этой строки для функции lodsb и stosb

L2: - Обнуление буффера и %edi = адрес буфера

L3: - Проверка на слово, точку, запятую.

make\_word: - создание слова в буфере и помещение в дерево

end\_of\_text: - завершение работы программы

init\_heap: - определение адреса heap

F1: - цикл для определения блоков памяти размером 48 байт

get\_free\_space: - взять кусок памяти

no\_free\_space: - если не существует такой кусок

create\_node: - создать узел для дерево

insert: - вставить этот узел для дерева

not\_null: - условие того, что дерево не пустое

ins\_loop: - помещаем слово в дерево. Смотрим: влево или вправо

less: - Левая ветка пустая? Если да, то пишем слово. Если нет – идем дальше

ins\_left: - Вставляем это слово в дерево

greater: - Правая ветка пустая? Если да, то пишем слово. Если нет – идем дальше

ins\_right: - Вставляем это слово в дерево

printTree: - Напечатать дерево

G0: - Условие того что дерево не пустое и печать «---» если слово встретилось впервые, печать

G1: - Печать слова слева, которое встретилось не впервые

G2: - Печать слова слева (Если лево больше не осталось, то начинаем аналогично справо)

Алгоритм работы программы:

1. Объявляем стек для дальнейшей работы
2. Создаем кучу
3. Начинаем ввод строки
4. Анализируем строку
5. Записываем слова в буфер
6. Вставляем слова в дерево
7. Печатаем дерево (С данными в задании условиями)
8. Завершаем программу

Заключение

Все требования к программе выполнены. Благодаря структуре дерева программа выполняется за время O(log n).

К сожалению, на некоторых версиях компилятора или операционной системы программа может сообщать об ошибке «Segmentation fault» при вызове функции printTree.

Не учитывая «кучу» и «строку ввода» программа достаточно эффективно использует выделенную память.

При тестировании программы критических ошибок, нарушающих работу операционной системы не обнаружено (Исключением является только печать дерева).

Список использованной литературы

1. Professional Assembly Language Richard Blum
2. Wikipedia

Приложения

1. Код программы с подробным описанием