МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Программирование»
Тема: Обзор стандартной библиотеки

Студентка гр. 0382 _____ Ситченко К.С.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2021

Цель работы.

Изучить и освоить возможности стандартной библиотеки языка Си.

Задание.

Вариант 1.

Напишите программу, на вход которой подается текст на английском языке (длина текста не превышает 1000 символов) и слово str (длина слова не превышает 30 знаков). Слова в тексте разделены пробелами или точкой. Программа должна вывести строку "exists", если str в тексте есть и "doesn't exist" в противном случае.

Программа должна реализовать следующий алгоритм:

- разбить текст на слова, используя функции стандартной библиотеки
- отсортировать слова, используя алгоритм быстрой сортировки (см. функции стандартной библиотеки)
- определить, присутствует ли в тексте str, используя алгоритм двоичного поиска (для реализации алгоритма двоичного поиска используйте функцию стандартной библиотеки)
- вывести строку "exists", если str в тексте есть и "doesn't exist" в противном случае.

Основные теоретические положения.

*void qsort(void *base, size_t num, size_t size, int (*compare) (const void *, const void *)) из stdlib.h*

Функция qsort() сортирует массив, на который указывает параметр base, используя quicksort — алгоритм сортировки широкого назначения, разработанный С. Р. Хори. После завершения функции массив становится отсортированным. Параметр num задает число элементов массива, параметр size задает размер в байтах каждого элемента. Функция, на которую указывает параметр compare, сравнивает элементы массива с ключом.

Формат функции compare следующий: *int func_name(const void *arg1, const void *arg2)*

Она должна возвращать следующие значения:

Если arg1 меньше, чем arg2, то возвращается значение меньше 0.

Если arg1 равно arg2, то возвращается 0.

Если arg1 больше, чем arg2, то возвращается величина больше 0.

Массив сортируется по возрастанию таким образом, что наименьший адрес соответствует наименьшему элементу.

*void *bsearch(const void *key, const void *base, size_t num, size_t size, int (*compare)(const void*, const void*)) из stdlib.h*

Функция bsearch() выполняет двоичный поиск на отсортированном массиве, на который указывает параметр base, и возвращает указатель на первое число, соответствующее ключу, на который указывает параметр key. Число элементов в массиве задается переменной num, а размер каждого элемента указывается в переменной size.

Тип size_t определен в заголовочном файле stdlib.h как unsigned int.

Функция, на которую указывает параметр compare, сравнивает элемент массива с ключом. Она должна иметь следующий прототип: *int func_name(const void *arg1, const void *arg2)*

Функция обязана возвращать следующие значения:

Если arg1 меньше, чем arg2, то возвращается величина, меньшая 0.

Если arg1 равен arg2, то возвращается величина 0.

Если arg1 больше, чем arg2, то возвращается число, большее 0.

Массив должен быть отсортирован по возрастанию таким образом, что наименьший адрес содержит наименьший элемент.

Если массив не содержит ключа, то возвращается нулевой указатель.

Выполнение работы.

Программа получает на вход текст (*char* text*) и слово (*char* str*), разбивает текст на отдельные слова (char** words) с помощью повторения функции *strtok*. Затем с использованием функции-компаратора (*int cmp(const void* a,const void* b)*) и функции быстрой сортировки *qsort* происходит сортировка массива слов, после чего посредством функции *bsearch* реализуется алгоритм двоичного поиска слова, происходит проверка условия (с помощью операторов *if, else*), её результат выводится на экран *("exists"* или *"doesn't exist"*). В конце программы освобождается выделенная память.

Разработанный программный код см. в приложении А.

Тестирование.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	Java is a general-purpose computer programming language that is concurrent class-based object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016 Java is one of the most popular programming languages in use particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems Java platform. is	exists	Программа сработала верно
2	Java is a general-purpose computer programming language that is concurrent class-based object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. qwe	doesn’t exist	Программа сработала верно

Выводы.

Были изучены и освоены возможности стандартной библиотеки языка Си.

Разработана программа, которая получает на вход текст, разбивает его на слова, сортирует, а затем выполняет поиск заданного слова, используя алгоритм двоичного поиска.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

1. Название файла: main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int cmp(const void* a,const void* b) {
  return strcmp(*(char**)a,*(char**)b);
}

int main(){
  char* text = malloc(1001*sizeof(char));
  fgets(text, 1001, stdin);
  text[strlen(text)] = '\0';
  char* str = malloc(31*sizeof(char));
  fgets(str, 31, stdin);
  str[strlen(str)] = '\0';

  int words_num = 20;
  char** words = calloc(words_num, sizeof(char*));
  int i = 0;
  char* ptr = strtok(text," .");
  while(ptr != NULL){
    words[i++] = ptr;
        if(i == words_num) {
          words_num*=2;
          words = realloc(words, words_num*sizeof(char*));
        }
        ptr = strtok(NULL," .");
  }

  qsort(words, i, sizeof(char**), cmp);
  char** ptr_str;
  ptr_str = bsearch(&str, words, i, sizeof(char**), cmp);
  if(ptr_str != NULL)
    printf("exists\n");
  else
    printf("doesn't exist\n");

  free(words);
  free(text);
  free(str);

  return 0;
}
```