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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование»
Тема: Использование указателей

Студент гр. 0382 Литягин С.М.

Преподаватель Чайка К.В., Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение и использование указателей в языке Си.

Задание.

Напишите программу, которая форматирует некоторый текст и выводит результат на консоль.

На вход программе подается текст, который заканчивается предложением "Dragon flew away!".

Предложение (кроме последнего) может заканчиваться на:

- . (точка)
- ; (точка с запятой)
- ? (вопросительный знак)

Программа должна изменить и вывести текст следующим образом:

- Каждое предложение должно начинаться с новой строки.
- Табуляция в начале предложения должна быть удалена.
- Все предложения, в которых есть цифра 7 (в любом месте, в том числе внутри слова), должны быть удалены.
- Текст должен заканчиваться фразой "Количество предложений до n и количество предложений после m", где n - количество предложений в изначальном тексте (**без учета** терминального предложения "Dragon flew away!") и m - количество предложений в отформатированном тексте (без учета предложения про количество из данного пункта).

Основные теоретические положения.

В программе использовались следующие управляющие конструкции языка:

Функции библиотеки *stdio.h*:

- *printf()* – функция вывода на консоль;
- *scanf()* – функция ввода данных из консоли.

- *getchar()* – функция возвращает следующий символ из стандартного потока ввода.

Функции для создания и работы с динамическим массивом:

- *malloc (void* malloc (size_t size))* - выделяет блок из size байт и возвращает указатель на начало этого блока
- *calloc (void* calloc (size_t num, size_t size))* - выделяет блок для num элементов, каждый из которых занимает size байт и инициализирует все биты выделеного блока нулями
- *realloc (void* realloc (void* ptr, size_t size))* - изменяет размер ранее выделенной области памяти на которую ссылается указатель ptr. Возвращает указатель на область памяти, измененного размера.
- *free (void free (void* ptr))* - высвобождает выделенную ранее память.

Циклы:

- *while(){}* – каждая итерация проверяет, выполняется ли условие в круглых скобках, если оно верно, то выполняется код в фигурных скобках, а если неверно, то происходит выход из цикла;
- *for(){<переменная>; <условие>; <выражение_1>}* – код в теле цикла будет исполнятся до тех пор, пока объявленная в цикле переменная будет удовлетворять условию цикла, выражение_1 каким-либо способом меняет значение этой переменной.

Операторы:

- *if(){} ... else{}* – если выполняется условия, указанное в круглых скобках, то выполняется код в фигурных скобках после if, иначе – в фигурных скобках после else (else не является обязательной частью конструкции)
- *switch(<переменная>){case x:… break; … default:…break;}* – от значения переменной в круглых скобках зависит, какой

кейс будет выполняться (например, если переменная имеет значение x – выполнится case x). Если же не будет кейса с таким значением, то выполнится код из блока default.

Функции:

- *<тип_функции> имя_функции(<аргумент_1>, ... , <argument_n>) {}* – при вызове данной функции в главной (main) функции выполняется код в фигурных скобках, а затем возвращает значение оператором return (если тип функции не void)

Выполнение работы.

Функция *char* sentence():*

В данной функции происходит объявление некоторых локальных переменных:

1. *size_s* типа *int*, ей присвоено значение 180. Создано для хранения размера выделяемой памяти динамического массива;
2. *index* типа *int*, ей присвоено значение 0. Является счетчиком для количества символов в предложении;
3. *sent* типа *char** - указатель на выделенную память при помощи функции *malloc* размером *size_s*. Создана для хранения предложения;
4. *c* типа *char*, ей присвоено значение ‘f’ (на данном шаге значение не имеет значения). Создано для хранения введенного в будущем символа.

Затем начинается цикл *while*, условием выхода из которого является введенные символы: “.”, “!”, “?”, “;”. В это цикле переменная *c* принимает символ, полученный с помощью функции *getchar()*. Далее элемент динамического массива *sent[index]* принимает значение переменной *c*. Затем счетчик index увеличивает значение на 1. Если значение *index* совпадает со значением *size_s*, то *size_s* увеличивает свое значение на 50, а выделенную память для *sent* увеличивают с помощью функции *realloc*. В самом конце

функции происходит удаление пробелов, табуляций и переносов в начале введенного предложения при помощи цикла *while*, условием выхода из которого является отсутствие вышеперечисленных знаков в первой ячейке динамического массива *sent* (*sent[0]*). В этом цикле происходит сдвиг элементов массива влево на 1 ячейку. В финале функция возвращает *sent*.

Функция *int seven(char* sent):*

В этой функции происходит поиск символа “7” среди элементов динамического массива *sent*. Если символ есть, то возвращает 1, иначе – 0.

Функция *int main():*

В данной функции происходит объявление некоторых переменных:

1. *text* типа *char*** - указатель на выделенную память при помощи функции *malloc* размером *size_t*. Создана для хранения текста;
2. *size_t* типа *int*, ей присвоено значение 20. Создано для хранения размера выделяемой памяти динамического массива;
3. *sent_before* и *sent_after* типа *int*, им присвоено значение 0. Являются счетчиком для количества предложений до и после обработки;
4. *sent* типа *char** - указатель.

Далее начинается бесконечный цикл while, из которого программа выходит при помощи *break*. В этом цикле переменной sent присваивается результат работы функции *sentence()*. Далее к значению переменной *sent_before* прибавляется 1. Если *seven(sent)* возвращает 1, то освобождается выделенная память для *sent*. Иначе элементу массива *text[sent_after]* присваивается значение *sent*, значение *sent_after* увеличивается на 1. И если значение *text[sent_after-1*] является строкой “Dragon flew away!”, то цикл заканчивается при помощи *break*. Если же *text[sent_after-1]* не является этой строкой, то происходит сравнение значения *sent_after* и *size_t*. Если оно совпадает, то значение *size_t* увеличивается на 20, а выделенную память для *text* увеличивают с помощью функции *realloc*.

После цикла *while* происходит вывод всех предложений, содержащихся в динамическом массиве *text*. Потом выводится предложение с количеством предложений до и после работы программы. В самом конце освобождается выделенная память с помощью функции *free*.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1	asdaskdkaskdjjsakdas. kalkdlaksldkakowkdo;an sdnann nnnnnawjndj! amsdmawsd. Dragon flew away!	asdaskdkaskdjjsakdas. kalkdlaksldkakowkdo; ansdnann nnnnnawjndj! amsdmawsd. Dragon flew away! Количество предложений до 4 и количество предложений после 4	Программа работает правильно
2	kllkkasdokawodkwa; ajijwijdjawjdwa. awd;awld;w7. Dragon flew away!	kllkkasdokawodkwa; ajijwijdjawjdwa. awd; awld; Dragon flew away! Количество предложений до 5 и количество предложений после 4	Программа работает правильно

Выводы.

В ходе работы были изучены указатели в языке Си. А также была разработана программа с использованием указателей и динамических массивов.

Код программы см. в Приложении А.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

char* sentence(){
    int size_s = 180;
    int index = 0;
    char* sent;
    char c = 'f';
    sent = malloc(size_s*sizeof(char));
    while(c != '.' && c != ';' && c != '?' && c != '!'){
      c = getchar();
      sent[index] = c;
      index++;
      if(index == size_s){
        size_s += 50;
        sent = realloc(sent, size_s);
      }
    }
    while(sent[0] == ' ' || sent[0] == '\t' || sent[0] == '\n'){
      for(int i = 0; i < index-1; i++){
        sent[i] = sent[i+1];
      }
      sent[index-1] = '\0';
    }
    return sent;
}


int seven(char* sent){
  for(int i = 0; i < strlen(sent); i++){
    if(sent[i] == '7'){
      return 1;
    }
  }
  return 0;
}

int main()
{
    char** text;
    int size_t = 20;
    int sent_before = 0;
    int sent_after = 0;
    char* sent;
    text = malloc(size_t*sizeof(char*));
    while(1){
      sent = sentence();
      sent_before++;
      if(seven(sent)){
        free(sent);
      }
```

```
      else{
        text[sent_after] = sent;
        sent_after++;
        if(!strcmp(text[sent_after-1],"Dragon flew away!")){
          break;
        }
      }
      if(sent_after == size_t){
        size_t += 20;
        text = realloc(text, size_t*sizeof(char*));
      }
    }
    for(int k = 0; k < sent_after ; k++)
        printf("%s\n",text[k]);
    printf("Количество предложений до %d и количество предложений
после %d", sent_before - 1, sent_after-1);
    for(int j = 0; j < sent_after ; j++){
        free(text[j]);
    }
    free(text);
    return 0;
}
```