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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Программирование»

Тема: «Создание make-файла»

Студентка гр. 7381 _____ Машина Ю. Д.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

Цель работы.

Создать проект, состоящий из пяти файлов: main.c, print_str.c, get_name.c, print_str.h, get_name.h.

- Файл get_name.c должен содержать **описание** функции, которая **считывает** из входного потока имя пользователя и возвращает его.
- Файл get_name.h должен содержать **прототип** функции, которая **считывает** из входного потока имя пользователя и возвращает его.
- Файл print_str.c должен содержать **описание** функции, которая **принимает** в качестве аргумента строку и выводит её (функция ничего не возвращает).
- Файл print_str.h должен содержать **прототип** функции, которая **принимает** в качестве аргумента строку и выводит её (функция ничего не возвращает).
- Файл main.c содержит главную функцию, которая вызывает функцию из файла get_name.h, добавляет к результату выполнения функции строку “Hello, ” и передает полученную строку в функцию вывода строки из print_str.h.

Создать Makefile для проекта.

Основные теоретические положения.

Заголовочные файлы, необходимые для создания проекта:

1 **<stdio.h>** - содержит прототип функции "void puts(const char* string)", выводящей в поток вывода строку string.

Синтаксис:
#include < stdio.h >
int puts (const char *s);

Аргументы:
s – указатель на строку, которую необходимо вывести.

Возвращаемое значение:
EOF - в случае ошибки.
Не отрицательное число, если вывод прошел успешно.

Описание:
Функция puts выводит строку в стандартный поток вывода. После вывода строки производится переход на новую строку (вывод символа «новая строка»). Символ конца строки (нулевой символ) не выводится.

2 **<string.h>**- содержит прототип функции "char* strncat(char* destptr,

char* srcptr, size_t num)".

Синтаксис:
#include < string.h >
char *strncat (char *destination, const char *append, size_t n);

Аргументы:
destination – указатель на массив в который будет добавлена строка.
append – указатель на массив из которого будет скопирована строка.
n – максимальная длина добавляемой строки.

Возвращаемое значение:
Функция возвращает указатель на массив, в который добавлена строка (destination).

Описание:
Функция strncat добавляет в строку, на которую указывает аргумент destination, строку, на которую указывает аргумент append, пока не встретится символ конца строки или пока не будет добавлено n символов.

Символ конца строки помещается в конце объединенных строк.

Если строки перекрываются, результат объединения будет не определен.

3 **<stdlib.h>**- содержит функции для выделения и освобождения памяти.

void free(void* ptrmem), void* malloc(size_t sizemem);

Синтаксис:
#include < stdlib.h >
void * malloc(size_t sizemem);

Аргументы:

sizemem - размер выделяемого блока памяти в байтах.

Возвращаемое значение:

Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда void*, поэтому это тип данных может быть приведен к желаемому типу данных.

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

Описание:

Функция malloc выделяет блок памяти, размером sizemem байт, и возвращает указатель на начало блока.

Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.

Синтаксис:
#include < stdlib.h >
void free(void * ptrmem);

Аргументы:

ptrmem - указатель на блок памяти, ранее выделенный функциями malloc, calloc или realloc, которую необходимо высвободить. Если в качестве аргумента передается нулевой указатель, никаких действий не происходит.

Возвращаемое значение:

Нет.

Описание:

Функция free освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или realloc освобождается. Освобожденная память может дальше использоваться программами или ОС.

Вывод:

Были изучены функции puts, strncat, malloc, free, способ сборки программы с использованием утилиты make, система контроля версий git.

Исходный код проекта:

- Файл "get_name.h":

```
char* get_name();
```

- Файл "get_name.c":

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


char* get_name(){
char* name =
(char*)malloc(80*sizeof(char));
int i = 0;
char ch;
while ((ch = getchar()) != '\n')
{
name[i] = ch;
i++;
}
name[i] = '\0';
return name;
}
```

- Файл "print_str.h":

```
void print_str(char* string);
```

- Файл print_str.c

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

```
void print_str(char *hello){
puts(hello);
}
```

- Файл "main.c":

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include"get_name.h"
#include"print_str.h"

int main(){
char hello[90] = "Hello, ";
char* result;
result = get_name();
print_str(strncat(hello, result, 80));
free(result);
return 0;
}
```

- Файл Makefile:

```
all: hello
hello: main.o print_str.o get_name.o
gcc -o a.out main.o print_str.o
get_name.o
main.o: main.c get_name.h print_str.h
gcc -c main.c
print_str.o: print_str.c print_str.h
gcc -c print_str.c
get_name.o: get_name.c get_name.h
gcc –c get_name.c
```