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

# ОТЧЕТ

**по лабораторной работе №1**

**по дисциплине «Программирование»**

**Тема: «Создание make-файла»**

Студент гр. 7381 _____ Судакова П. С.

Преподаватель _____ Берленко Т. А.

Санкт-Петербург

2017

# Цель работы:

Создание проекта, состоящего из пяти файлов: `main.c, print_str.c, get_name.c, print_str.h, get_name.h.`

- Файл `get_name.c` содержащий **описание** функции, которая **считывает** из входного потока имя пользователя и возвращает его.
- Файл `get_name.h` содержащий **прототип** функции, которая **считывает** из входного потока имя пользователя и возвращает его.
- Файл `print_str.c` содержащий **описание** функции, которая **принимает** в качестве аргумента строку и выводит её (функция ничего не возвращает).
- Файл `print_str.h` содержащий **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.`

После выполнения проекта, создать для него `Makefile`, с помощью которого данный проект будет собираться.

# Основные теоретические положения:

**Заголовочные файлы языка С, необходимые для выполнения лабораторной работы:**

**1**. Функция «`void print_str(char* string)`». В этой функции заголовочный файл «`stdio.h`» ( стандартный заголовочный файл ввода-вывода) подключает прототип функции «`int puts(const char *str)`», которая склеивает строку приветствия и имя.

**Описание:**

Функция «`puts`» выводит строку типа «`char*`», на которую указывает параметр «`string`» в стандартный поток вывод и добавляет символ новой строки «'`n`'».

Функция начинает копировать строку с адреса, указанного в `string`, пока не достигнет нулевого символа ". Этот заключительный, нулевой символ не копируется в стандартный поток вывод.

**Параметры:**
«`const char* string`» - С-строка для вывода на стандартный поток вывода.

**Возвращаемое значение:**
В случае успеха, возвращается неотрицательное значение.
В случае ошибки, функция возвращает значение EOF.

**2.** Функция «`int main()`». В этой функции заголовочный файл «`string.h`» подключает прототип функции «`char * strncat( char * destptr, char * srcptr, size_t num )`», которая склеивает строку приветствия и имя.

**Описание:**
Функция добавляет первые «`num`» символов строки «`srcptr`» к концу строки «`destptr`», плюс символ конца строки. Если строка «`srcptr`» больше чем количество копируемых символов «`num`», то после скопированных символов неявно добавляется символ конца строки**.**

**Параметры:** «`destptr`» - Указатель на строку назначения, которая будет содержать результат конкатенации строк, включая символ завершения строки.
«srcptr» - Строка, из которой будут копироваться первые «num» символов для конкатенации.
«num» - Максимальное количество символов для конкатенации.

**Возвращаемое значение:**
Указатель на строку с результатом конкатенации.

**3**. Функция «`char* get_name()`». В этой функции заголовочный файл <`stdlib.h`> - содержит функции для выделения и освобождения памяти.

- `void free(void* ptrmem);`

**Описание:**
Функция `free` освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова `malloc, calloc` или `realloc` освобождается. То
есть освобожденная память может дальше использоваться программами или ОС.

**Параметры:**

`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:**

Функция не имеет возвращаемое значение.

- `void* malloc(size_t sizemem);`

**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.

**Параметры:**

`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.

# Исходный код проекта:

- **Файл «get_name.h»**

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

- **Файл «get_name.c»**

```
#include <stdlib.h>
#include <stdio.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»**

```
int print_str(const char *str);
```

- **Файл «print_str.c»**

```
#include "stdio.h"
int print_str(const char *str){
puts(str);
return 0;
}
```

- **Файл «main.c»**

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

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

- **Makefile**

```
all: main.o print_str.o get_name.o
    gcc main.o print_str.o get_name.o
main.o: main.c print_str.h get_name.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
clean:
    rm -rf *.o hello
```

# Вывод:

В ходе лабораторной работы был освоен терминал ОС Linux и его элементарные функции (создание, удаление, редактирование файлов и директорий, просмотр их содержимого). Так же были изучены команды, позволяющие использовать систему контроля версий "Git" (git clone, git push, git commit, git status, git diff, pull, merge). В свою очередь, была освоена база языка C и принцип работы с make-файлами.