

13 Лабораторная работа

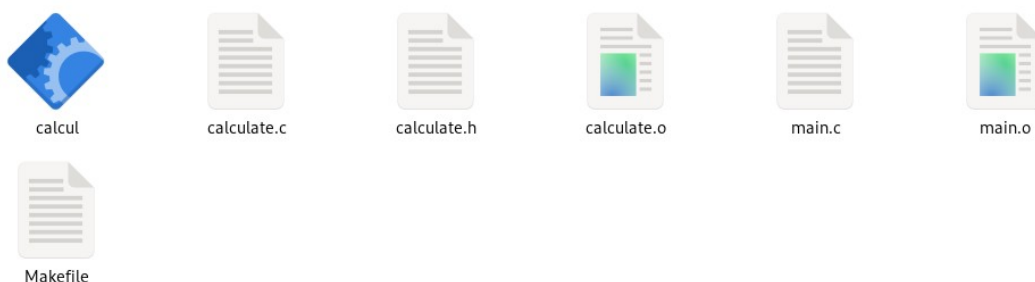
Прищепов Александр

Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

Выполнение лабораторной работы

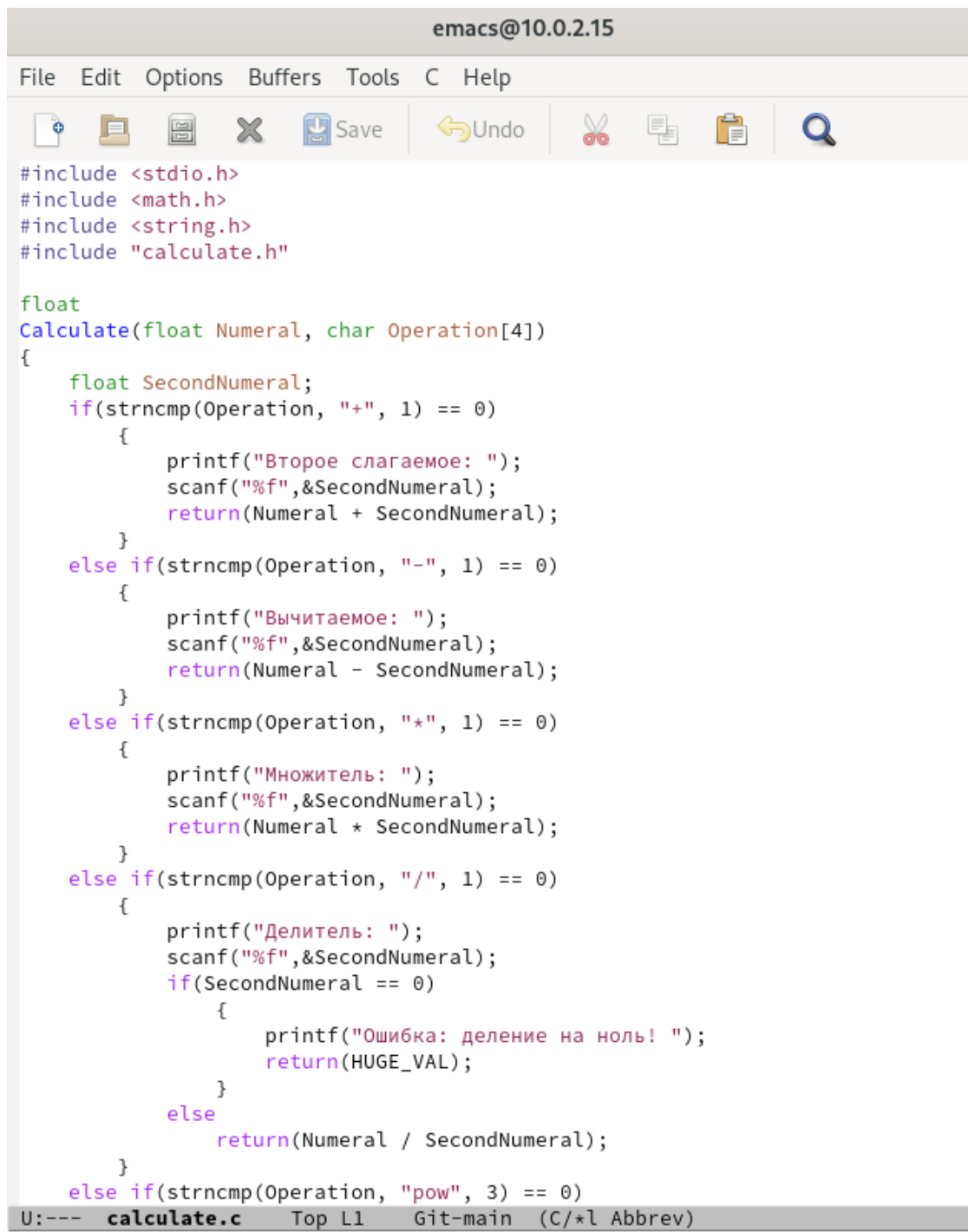
1. В домашнем каталоге создаем подкаталог `~/work/os/lab_prog` и в нем уже создаем три файла: `calculate.h`, `calculate.c`, `main.c` (рис. 1). Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять `sin`, `cos`, `tan`. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится.



изображение

2. В созданных файлах напишем программы для работы калькулятора, которые нам предоставили (рис. 2), (рис. 3), (рис. 4).

рис 2:



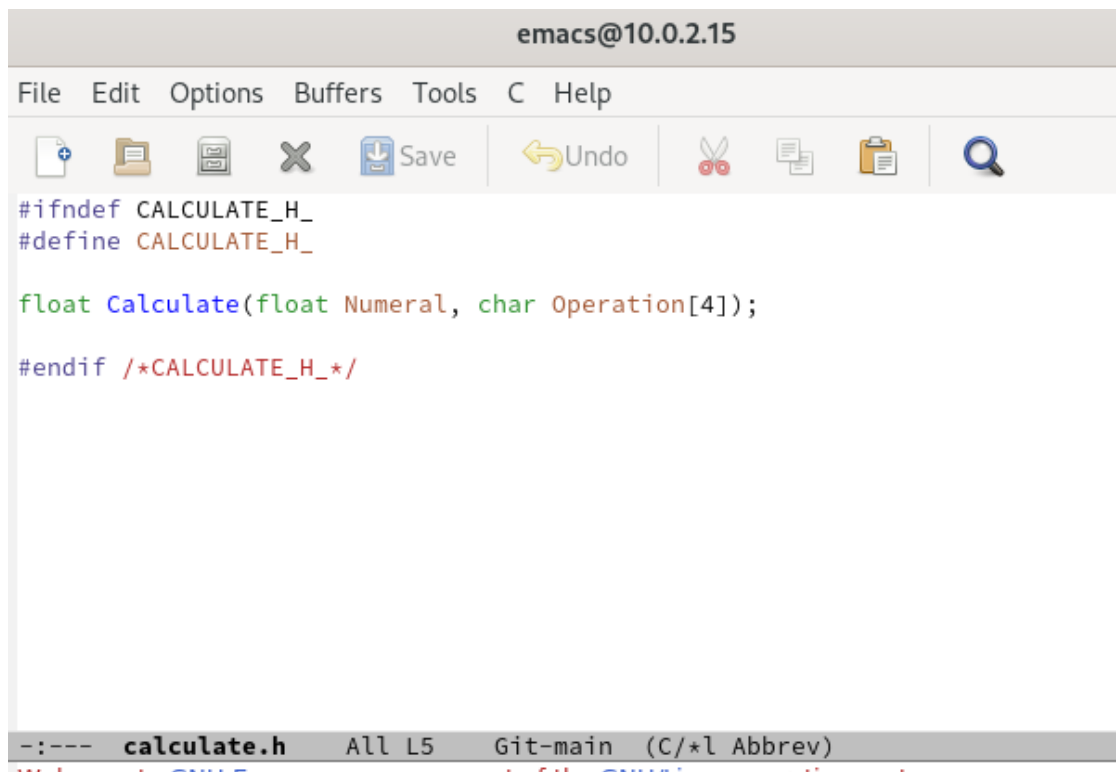
```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "calculate.h"

float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if(strncmp(Operation, "+", 1) == 0)
    {
        printf("Второе слагаемое: ");
        scanf("%f",&SecondNumeral);
        return(Numeral + SecondNumeral);
    }
    else if(strncmp(Operation, "-", 1) == 0)
    {
        printf("Вычитаемое: ");
        scanf("%f",&SecondNumeral);
        return(Numeral - SecondNumeral);
    }
    else if(strncmp(Operation, "*", 1) == 0)
    {
        printf("Множитель: ");
        scanf("%f",&SecondNumeral);
        return(Numeral * SecondNumeral);
    }
    else if(strncmp(Operation, "/", 1) == 0)
    {
        printf("Делитель: ");
        scanf("%f",&SecondNumeral);
        if(SecondNumeral == 0)
        {
            printf("Ошибка: деление на ноль! ");
            return(HUGE_VAL);
        }
        else
            return(Numeral / SecondNumeral);
    }
    else if(strncmp(Operation, "pow", 3) == 0)
    {
        printf("Повышение в степень: ");
        scanf("%f",&SecondNumeral);
        return(pow(Numeral, SecondNumeral));
    }
}
```

U:--- calculate.c Top L1 Git-main (C/*l Abbrev)

изображение

рис 3:



The screenshot shows the Emacs editor window titled "emacs@10.0.2.15". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "C", and "Help". The toolbar contains icons for opening files, saving, undo, redo, and search. The code in the buffer is as follows:

```
#ifndef CALCULATE_H_
#define CALCULATE_H_

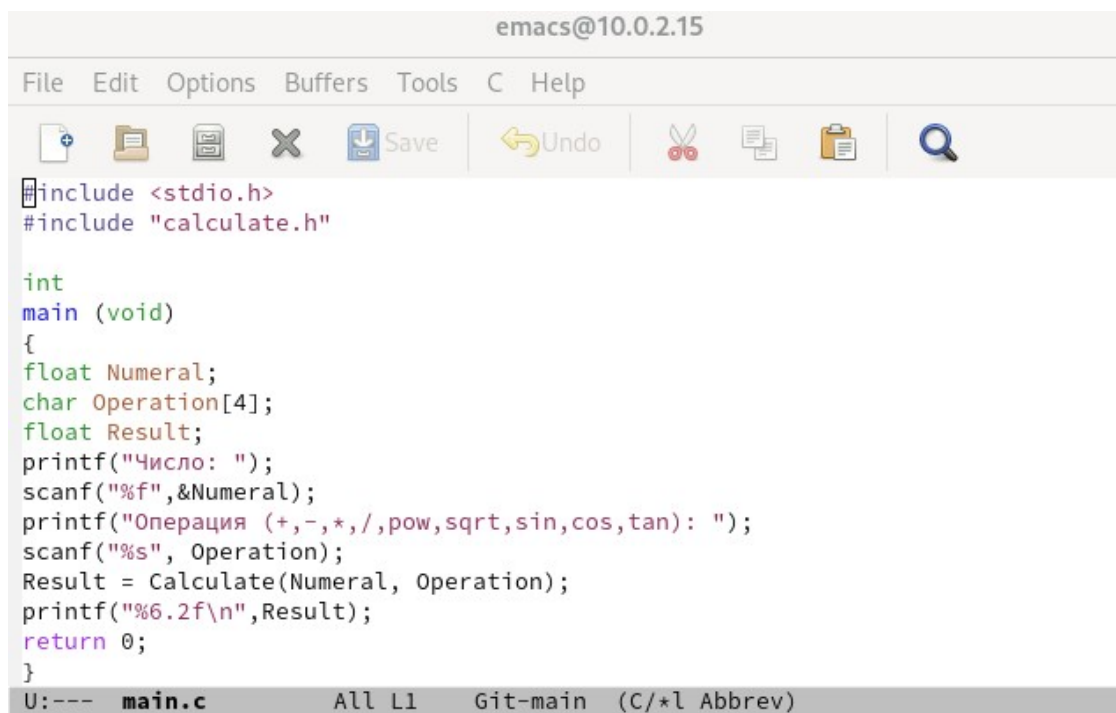
float Calculate(float Numeral, char Operation[4]);

#endif /*CALCULATE_H_*/
```

The status bar at the bottom indicates the current file is "calculate.h", the buffer is "All L5", and the repository is "Git-main (C/*l Abbrev)".

изображение

рис 4:



The screenshot shows the Emacs editor window titled "emacs@10.0.2.15". The menu bar and toolbar are the same as in the previous image. The code in the buffer is as follows:

```
#include <stdio.h>
#include "calculate.h"

int
main (void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Число: ");
    scanf("%f",&Numeral);
    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
    scanf("%s", Operation);
    Result = Calculate(Numeral, Operation);
    printf("%6.2f\n",Result);
    return 0;
}
```

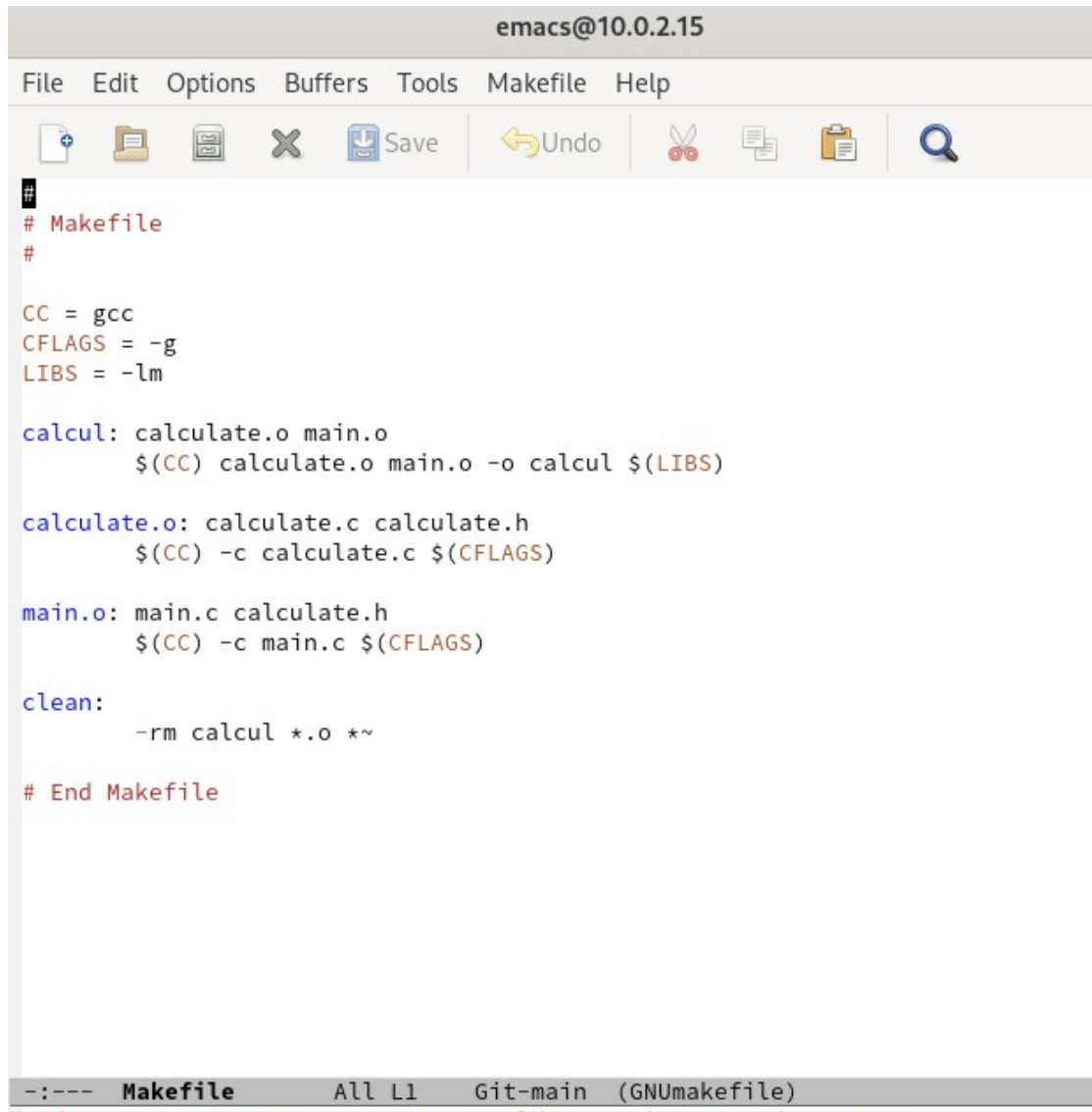
The status bar at the bottom indicates the current file is "main.c", the buffer is "All L1", and the repository is "Git-main (C/*l Abbrev)".

изображение

3. Выполним компиляцию программы посредством gcc и при необходимости исправим синтаксические ошибки

4. Создадим Makefile и введем в него предложенное содержимое (рис. 5).

рис 5:



```
#
# Makefile
#

CC = gcc
CFLAGS = -g
LIBS = -lm

calcul: calculate.o main.o
    $(CC) calculate.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
    $(CC) -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
    $(CC) -c main.c $(CFLAGS)

clean:
    -rm calcul *.o *~

# End Makefile
```

изображение

Данный файл необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а также их объединения в один исполняемый файл calcul (цель calcul). Цель clean нужна для автоматического удаления файлов. Переменная CC отвечает за утилиту для компиляции. Переменная CFLAGS отвечает за опции в данной утилите. Переменная LIBS отвечает за опции для объединения объектных файлов в один исполняемый файл.

5. Далее исправим Makefile. В переменную CFLAGS добавил опцию -g, необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделаем так, что утилита компиляции выбирается с помощью переменной CC.

После этого удалим исполняемые и объектные файлы из каталога с помощью команды `make clean`. Выполним компиляцию файлов, используя команды `make calculate.o`, `make main.o`, `make calcul`.

6. Далее с помощью команды `gdb ./calcul` запустим отладку программы
 - Для запуска программы внутри отладчика введем команду `run`
 - Для постраничного (по 9 строк) просмотра исходного кода используем команду `list`
 - Для просмотра строк с 12 по 15 основного файла используем `list` с параметрами
 - Для просмотра определённых строк не основного файла используем `list` с параметрами
 - Установим точку останова в файле `calculate.c` на строке номер 18 и выведем информацию об имеющихся в проекте точке останова
 - Запустим программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова
 - Введем команду `backtrace`, которая покажет весь стек вызываемых функций от начала программы до текущего места
 - Посмотрим, чему равно на этом этапе значение переменной `Numeral`, введя команду `print Numeral` и сравним с результатом команды `display Numeral`
 - Уберем точки останова

7. С помощью утилиты splint проанализируем коды файлов `calculate.c` и `main.c`. Воспользуемся командами `splint calculate.c` и `splint main.c`.

С помощью утилиты `splint` выяснилось, что в файлах `calculate.c` и `main.c` присутствует функция чтения `scanf`, возвращающая целое число (тип `int`), но эти числа не используются и нигде не сохраняются. Утилита вывела предупреждение о том, что в файле `calculate.c` происходит сравнение вещественного числа с нулем. Также возвращаемые значения (тип `double`) в функциях `pow`, `sqrt`, `sin`, `cos` и `tan` записываются в переменную типа `float`, что свидетельствует о потере данных.

Выводы

Здесь кратко описываются итоги проделанной работы.