1. Установка программного обеспечения.

ПО устанавливается на компьютер с операционной системой Windows 10 64bit

- На официальном сайте mingw.org был скачан установщик. Затем в процессе установки были выбраны нужные компоненты и необходимые пакеты. Изменена переменная окружения PATH. В ней указан путь к папке *C:\MinGW\bin;C:\MinGW\msys\1.0\bin*
- *С помощью инсталлятора с официального сайта произведена on-line установка библиотеки Qt, в которой Qt Creator версии 3.5.1 является одной из устанавливаемых компонент.*

Проблем с установкой и настройкой ПО не возникло.

2. Этапы компиляции программы

a. *Код программы*

файл triangle.c (544 байт)

```c
#include <stdio.h>
#include <math.h>
#define P(A, B, C) (A+B+C)/2  // подсчет полупериметра

int main(void)
{
    float a,b,c;
    float area;
    puts("Input a,b,c");
    int counter = scanf("%f %f %f",&a,&b,&c);
    // проверка ввода
    if (counter == 3)
    {
        // формула Герона вычисления площади треугольника
        area = sqrt(P(a,b,c)*(P(a,b,c)-a)*(P(a,b,c)-b)*(P(a,b,c)-c));
        printf("Area of triangle: %6.2f",area);
    }
    else
    {
        puts("Incorrect entry");
    }
}
```

b. *Этапы компиляции*

Предпроцессор

Команда: cpp -o triangle.i triangle.c

Результат: получение файла triangle.i (33938 байт)

```c
int main(void)
{
 float a,b,c;
 float area;
 puts("Input a,b,c");
 int counter = scanf("%f %f %f",&a,&b,&c);

 if (counter == 3)
 {

  area = sqrt((a+b+c)/2*((a+b+c)/2 -a)*((a+b+c)/2 -b)*((a+b+c)/2 -c));
  printf("Area of triangle: %6.2f",area);
 }
 else
 {
  puts("Incorrect entry");
 }
}
```

Трансляция на ассемблер

Команда: c99 -S -masm=intel triangle.i

Результат: получение файла triangle.s (6683 байт)

Часть кода, содержащая функцию main

```
_main:
LFB10:
        .cfi_startproc
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
        .cfi_def_cfa_register 5
        and     esp, -16
        sub     esp, 64
        call    ___main
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _puts
        lea     eax, [esp+44]
        mov     DWORD PTR [esp+12], eax
        lea     eax, [esp+48]
        mov     DWORD PTR [esp+8], eax
        lea     eax, [esp+52]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC1
        call    _scanf
        mov     DWORD PTR [esp+60], eax
        cmp     DWORD PTR [esp+60], 3
        jne     L32
        fld     DWORD PTR [esp+52]
        fld     DWORD PTR [esp+48]
        faddp   st(1), st
        fld     DWORD PTR [esp+44]
        faddp   st(1), st
        fld     TBYTE PTR LC2
        fdivp   st(1), st
        fld     DWORD PTR [esp+52]
        fld     DWORD PTR [esp+48]
        faddp   st(1), st
        fld     DWORD PTR [esp+44]
        faddp   st(1), st
        fld     TBYTE PTR LC2
        fdivp   st(1), st
        fld     DWORD PTR [esp+52]
        fsubp   st(1), st
        fmulp   st(1), st
        fld     DWORD PTR [esp+52]
        fld     DWORD PTR [esp+48]
        faddp   st(1), st
        fld     DWORD PTR [esp+44]
        faddp   st(1), st
        fld     TBYTE PTR LC2
        fdivp   st(1), st
        fld     DWORD PTR [esp+44]
        fsubp   st(1), st
        fmulp   st(1), st
        fld     DWORD PTR [esp+52]
        fld     DWORD PTR [esp+48]
        faddp   st(1), st
        fld     DWORD PTR [esp+44]
        faddp   st(1), st
        fld     TBYTE PTR LC2
        fdivp   st(1), st
        fld     DWORD PTR [esp+44]
        fsubp   st(1), st
        fmulp   st(1), st
        fstp    QWORD PTR [esp+24]
        fld     QWORD PTR [esp+24]
        fstp    QWORD PTR [esp]
        call    _sqrt
        fstp    DWORD PTR [esp+56]
        fld     DWORD PTR [esp+56]
        fstp    QWORD PTR [esp+4]
        mov     DWORD PTR [esp], OFFSET FLAT:LC3
        call    _printf
        jmp     L33
L32:
        mov     DWORD PTR [esp], OFFSET FLAT:LC4
        call    _puts
L33:
        mov     eax, 0
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
```

Часть кода с вводом и выводом текста

```
LC0:
        .ascii "Input a,b,c\0"
LC1:
        .ascii "%f %f %f\0"
LC3:
        .ascii "Area of triangle: %6.2f\0"
LC4:
        .ascii "Incorrect entry\0"
```

Ассемблирование в объектный файл

Команда: as -o triangle.o triangle.s

Результат: получение объектного файла triangle.o (2290 байт)

Компоновка

Команда: gcc -o triangle.exe triangle.o

Результат: получение исполняемого файла triangle.exe (78313 байт)

3. Исследование исполняемого файла

c. Секции в программе с/без отладочной информации

Сборка без отладочной информации

Команды: gcc -std=c99 -Wall -Werror -o triangle_wo.exe triangle.c
objdump -x triangle_wo.exe > wo.txt

Размер файла: 69266 байт

Секции:

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00007504  00401000  00401000  00000400  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
  1 .data         00000028  00409000  00409000  00007a00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .rdata        00000850  0040a000  0040a000  00007c00  2**5
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .eh_fram      000015e4  0040b000  0040b000  00008600  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .bss          00000a34  0040d000  0040d000  00000000  2**5
                  ALLOC
  5 .idata        000006d0  0040e000  0040e000  00009c00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  6 .CRT          00000018  0040f000  0040f000  0000a400  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  7 .tls          00000020  00410000  00410000  0000a600  2**2
                  CONTENTS, ALLOC, LOAD, DATA
```

Сборка с отладочной информацией

Команды: gcc -std=c99 -Wall -Werror -o triangle_w.exe triangle.c
objdump -x triangle_w.exe > w.txt

Размер файла: 83924 байт

Секции:

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00007504  00401000  00401000  00000400  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
  1 .data         00000028  00409000  00409000  00007a00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .rdata        00000850  0040a000  0040a000  00007c00  2**5
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .eh_frame     000015e4  0040b000  0040b000  00008600  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .bss          00000a34  0040d000  0040d000  00000000  2**5
                  ALLOC
  5 .idata        000006d0  0040e000  0040e000  00009c00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  6 .CRT          00000018  0040f000  0040f000  0000a400  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  7 .tls          00000020  00410000  00410000  0000a600  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  8 .debug_aranges 00000020  00411000  00411000  0000a800  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_info   0000028a  00412000  00412000  0000aa00  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_abbrev 000000e0  00413000  00413000  0000ae00  2**0
                  CONTENTS, READONLY, DEBUGGING
 11 .debug_line   000000c6  00414000  00414000  0000b000  2**0
                  CONTENTS, READONLY, DEBUGGING
 12 .debug_macro  00002d33  00415000  00415000  0000b200  2**0
                  CONTENTS, READONLY, DEBUGGING
```

d. В какие секции попадают переменные и функции

Добавим в текст программы неинициализированную переменную.
С помощью утилиты objdump сравним секции до добавления переменной и после

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\compile\sections>fc out1.txt out2.txt
Сравнение файлов out1.txt и OUT2.TXT
***** out1.txt
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .bss          00000a34  0040d000  0040d000  00000000  2**5
                  ALLOC
***** OUT2.TXT
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .bss          00000a38  0040d000  0040d000  00000000  2**5
                  ALLOC
*****
```

Следовательно, неициализированная переменная попала в секцию .bss

Добавим в текст программы инициализированную переменную.
С помощью утилиты objdump сравним секции до добавления и после

```
Сравнение файлов out1.txt и OUT3.TXT
***** out1.txt
                  CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
  1 .data         00000028  00409000  00409000  00007a00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
***** OUT3.TXT
                  CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
  1 .data         0000002c  00409000  00409000  00007a00  2**2
                  CONTENTS, ALLOC, LOAD, DATA
*****
```

Инициализированная переменная попадает в секцию .data

Добавим в текст программы функцию

```
int func()
```

С помощью утилиты objdump сравним секции до добавления и после

```
Сравнение файлов out1.txt и OUT5.TXT
***** out1.txt
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00007504  00401000  00401000  00000400  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
***** OUT5.TXT
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00007524  00401000  00401000  00000400  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE, DATA
*****
```

Функции попадают в секцию .text

e. Добавление глобального неициализированного массива.

Добавим глобальный неициализированный массив

```
int array[3];
```

Сравним вывод objdump до добавления и после

```
Сравнение файлов out1.txt и OUT2.TXT
***** out1.txt
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .bss          00000a34  0040d000  0040d000  00000000  2**5
                  ALLOC
***** OUT2.TXT
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  4 .bss          00000a40  0040d000  0040d000  00000000  2**5
                  ALLOC
*****
```

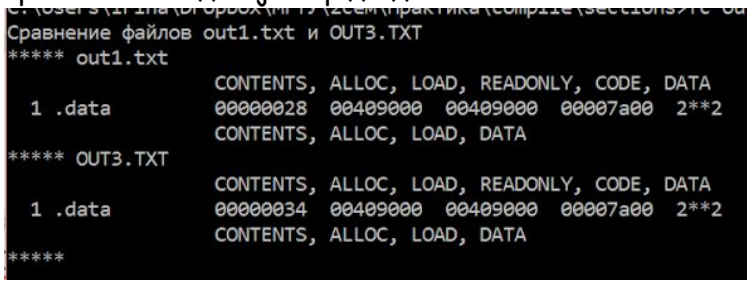
Неициализированный массив попал в секцию .bss .
Размер исполняемого файла до добавления массива - 69266 байт
Размер исполняемого файла после добавления массива - 69284 байт

f. Добавление глобального инициализированного массива.

Добавим глобальный неициализированный массив

```
int array[3]={1,2,3};
```

Сравним вывод objdump до добавления и после



Инициализированный массив попал в секцию .data .

Размер исполняемого файла до добавления массива - 69266 байт

Размер исполняемого файла после добавления массива - 69284 байт

g. Используемые динамические библиотеки

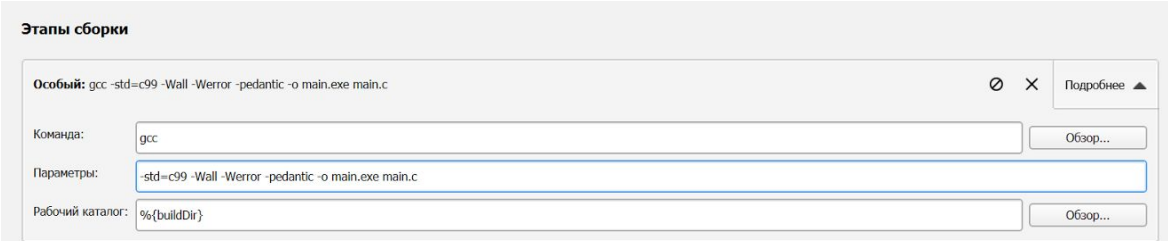
Команда: objdump -p triangle.exe

```
...
DLL Name: KERNEL32.dll
...
DLL Name: msvcrt.dll
...
```

4. Работа с QT Creator

b. *Проект для программы*

Для компиляции и сборки программы



c. *Исправление ошибок*

1. ошибка: stdoi.h: No such file or directory

```
#include <stdoi.h>
            ^
```

#include <stdoi.h> -> #include <stdio.h>

2. ошибка: format '%f' expects argument of type 'float *', but argument 2 has type 'int *' [-Werror=format=]

```
   scanf("%f", &max);
   ^
```

scanf("%f", &max); -> scanf("%d", &max);

3. ошибка: 'num' undeclared (first use in this function)

```
   while (scanf("%d", num) == 1)
                ^
```

```
int num;
   while (scanf("%d", num) == 1) ->      while (scanf("%d", &num) == 1)
```

4. ошибка: suggest parentheses around assignment used as truth value [-Werror=parentheses]

```
        if (num = max)
        ^
```

if (num = max) -> if (num == max)

5. printf("max %d, count %d\n", &max, &count); ->
printf("max %d, count %d\n", max, count);

6. count = 0; -> count = 1;

7.

```
if (num > max)        ->    if (num > max)
    max = num;                {
                                  max = num;
                                  count = 1;
                              }
```

8. Не рассматриваются случаи, когда чисел нет

```
if (scanf("%d", &max) == 0)
     puts("No numbers");
  else
  { … }
```

d.

хз как это делается гугл не помог

e.

хз как это делается гугл не помог

f.

хз как это делается гугл не помог

5. Индивидуальное задание

Задача 1.

Условие

Пользователь вводит целые числа, по окончании ввода чисел нажимает Ctrl-Z и Enter. Написать программу, которая определяет сколько раз в последовательности чисел меняется знак (нуль считается положительным числом).

Допущения

- 0 - положительное число
- Если число одно - будем считать, что знак меняется 0 раз

Алгоритм

главная функция

```
целое first
целое second
целое s
сообщение("Input numbers:")
считать first, если это число
в противном случае сообщение("No numbers")
считывать second пока это число
    s = s + counter(first,second)
    first = second
сообщение("Mark of sequence is changing",s,"times")
```

функция counter(целое x, целое y)

```
если x*y < 0
    возврат 1
если x*y > 0
    возврат 0
если x*y = 0
    если (x = 0 и y < 0) или (y = 0 и x > 0)
        возврат 1
    в противном случае
        возврат 0
```

Классы эквивалентности для тестовых данных

1. Некорректный ввод
2. Одно число
3. Все числа одного знака
4. Имеются числа разных знаков
5. В последовательности чисел присутствует 0

Примеры

1. *Некорректный ввод*
   Ctrl+Z
   abc
   abc -5 5
2. *Одно число*
   5
   -5 abc
   5 abc -5
3. *Все числа одного знака*
   1 2 3 4
   -1 -2 -3 -4
4. *Имеются числа разных знаков*
   1 -2 -3 4
   -5 -6 7 8
5.
   1 0 -1
   -5 0

Задача 2.

Условие

Написать программу, которая считывает из текстового файла вещественные числа и выполняет над ними некоторые вычисления: найти число, наиболее близкое к среднему значению всех чисел.

Допущения

- Файл содержит только числа
- Если наиболее близких к среднему значению чисел несколько, то выводится первое

Алгоритм

```
главная функция
    вещественное avrg
    avrg = average(file)
    если avrg <> -1
        сообщение ("Average is",avrg)
        сообщение ("Close to average number of sequence is". search(file,avrg))
    в противном случае
        сообщение("No numbers in file")

функция average(файл file) возвращает вещественное число
    целое count
    вещественное sum
    вещественное num
    считать num если это число
        sum = num
        count = 1
        считать num пока это число
            sum = sum + num
            count = count + 1
        возврат sum/count
    в противном случае
        возврат -1
```

```
функция search(файл file, вещественное avrg) возвращает вещественное число
        вещественное minmod
        вещественное need
        вещественное num
        считать num если это число
            minmod = mod(num-avrg)
            need = num
        считать num пока это число
            если mod(num-avrg) < minmod
            minmod = mod(num-avrg)
                    need = num
    возврат need
```

Классы эквивалентности для тестовых данных

1. Чисел в файле нет
2. Число в файле одно
3. В файле несколько чисел одинаково близких к среднему значению
4. В файле одно число, значение которого наиболее близко к среднему значению

Примеры

1. Чисел в файле нет

2. Число в файле одно
   5
3. В файле несколько чисел одинаково близких к среднему значению
   1 2 3 5 6 7
   2.2 2.4
4. В файле одно число, значение которого наиболее близко к среднему значению
   1 6 4 8 7
   -1.1 2.3 4.2

6. Исследование покрытия кода тестами

Задание 1.

Команда :

c99 -Wall -Werror -pedantic -O0 -fprofile-arcs -ftest-coverage 1.c -o 1.exe

Тест 1 - пустой ввод. Покрытие 42.31%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>1.exe < test1.txt
Input numbers:
Result in file 'out.txt'
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>gcov 1.c
File '1.c'
Lines executed:42.31% of 26
Creating '1.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:100.00% of 10
Creating 'stdio.h.gcov'
```

Тест 2 - одно число в файле. Покрытие 61.54%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>1.exe < test2.txt
Input numbers:
Result in file 'out.txt'
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>gcov 1.c
File '1.c'
Lines executed:61.54% of 26
Creating '1.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:100.00% of 10
Creating 'stdio.h.gcov'
```

Тест 3 - все числа одного знака. Покрытие 84.62%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>1.exe < test3.txt
Input numbers:
Result in file 'out.txt'
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>gcov 1.c
File '1.c'
Lines executed:84.62% of 26
Creating '1.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:100.00% of 10
Creating 'stdio.h.gcov'
```

Тест 4 - имеются числа разных знаков. Покрытие 88.46%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>1.exe < test4.txt
Input numbers:
Result in file 'out.txt'
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>gcov 1.c
File '1.c'
Lines executed:88.46% of 26
Creating '1.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:100.00% of 10
Creating 'stdio.h.gcov'
```

Тест 5 **-** в файле присутствует 0. Покрытие 100.00%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>1.exe < test5.txt
Input numbers:
Result in file 'out.txt'
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\1\first>gcov 1.c
File '1.c'
Lines executed:100.00% of 26
Creating '1.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:100.00% of 10
Creating 'stdio.h.gcov'
```

Результат - получение файла 1.c.gcov

Задание 2.

Команда:
c99 -Wall -Werror -pedantic -O0 -fprofile-arcs -ftest-coverage 2.c func.c -o 2.exe

Тест 1 *- пустой ввод*
Тест 2 *- одно число*
Тест 3 - несколько чисел одинаково близких к среднему значению
Тест 4 - одно число, значение которого наиболее близко к среднему значению
Покрытие 75.00%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>2.exe test1.txt
No numbers in file
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>2.exe test2.txt
Average is 1.00
Close to average number of sequence is 1.00
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>2.exe test3.txt
Average is 4.00
Close to average number of sequence is 3.00
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>2.exe test4.txt
Average is 5.20
Close to average number of sequence is 6.00
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>gcov 2.exe
File '2.c'
Lines executed:75.00% of 16
Creating '2.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:50.00% of 10
Creating 'stdio.h.gcov'
```

Тест 5 - неверное имя файла/запрашиваемый файл не создан
Тест 6 - неверно задано имя файла как параметра
Покрытие 100.00%

```
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>2.exe test.txt
File doesn't exist
C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>2.exe test1.txt test2.txt
Put the name of file as parameter of launch

C:\Users\Irina\Dropbox\МГТУ\2сем\Практика\indtask\2\second>gcov 2.exe
File '2.c'
Lines executed:100.00% of 16
Creating '2.c.gcov'

File '<built-in>'
No executable lines
Removing '<built-in>.gcov'

File 'c:/mingw/include/stdio.h'
Lines executed:100.00% of 10
Creating 'stdio.h.gcov'
```

Результат - получение файла 2.c.gcov

Заключение

- Существует **4 этапа компиляции** программы - работа предпроцессора, трансляция на ассемблер, ассемблирование в объектный файл, компоновка. Проект можно собрать из командной строки поэтапно.
- С помощью утилиты **objdump** можно получить представление об организации объектных и исполняемых файлов и анализировать содержащуюся в них информацию.
- С помощью **интегрированной среды разработки Qt Creator** можно создавать проекты, настраивать сборки проектов, отлаживать проекты.
- С помощью утилиты **make** можно автоматизировать сборки проектов как в командной строке, так и в среде Qt Creator
- Утилита **gcov** позволяет определить величину покрытия кода тестами