Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторной работе

Дисциплина: Низкоуровневое программирование

Тема: Раздельная компиляция

Выполнил студент гр. 3530901/10005 _____
Калашников О. Ю.

(подпись)

Преподаватель _____
Коренев Д.А.

(подпись)

“___”_____ 2022 г.

Санкт-Петербург

2022

Оглавление

ТЗ

Написать программу на языке С, которая реализует нахождение максимального элемента массива. Также поместить определение функции в отдельный исходный файл, оформить заголовочный файл и разработать тестовую программу на языке С.

Пошагово собрать программу. Проанализировать выход препроцессора и компилятора. Проанализировать состав и содержимое секций, таблицы символов, таблицы перемещений и отладочную информацию, содержащуюся в объектных файлах и
исполняеммом файле.

Выделить разработанную функцию в статическую библиотеку. Разработать Makefile для сборки библиотеки и использующей ее тестовой программы. Проанализировать ход сборки библиотеки и программы, созданные файлы зависимостей.

Метод решения

Для нахождения максимального элемента массива необходимо циклически пройти каждый элемент, сравнивая его с максимальным значением, найденным на данный момент (для первого элемента текущее максимальное значение принимается за 0).

Например, рассмотрим массив [3, 5, 9, 4, 1]:
1. 3 > 0 => 3 — текущий максимальный элемент;
2. 5 > 3 => 5 — текущий максимальный элемент;
3. 9 > 5 => 9 — текущий максимальный элемент;
4. 4 < 9 => 9 — остаётся текущим максимальным элементом;
5. 1 < 9 => 9 — остаётся текущим максимальным элементом;
Ответ: 9.

Реализация программ на языке C

Тестовая программа

main.c > main(void)

```c
#include <stdio.h>
#include "maxnum.h"

int main(void) {
    int array[] = {10, 5, 3, 2, 12, 69, 35, 69, 70, 11};
    int array_length = sizeof(array)/sizeof(array[0]);
    printf("%d", maxNum(array, array_length));
    return 0;
    }

```

Функциональная часть

maxnum.c > maxNum(int [], int)

```c
#include <stdio.h>
#include "maxnum.h"

int maxNum(int array[], int array_length) {
    int max = array[0];
    for (int i = 0; i < array_length; i++) {
        if (array[i] > max)
        {
            max = array[i];
        }
    }
    return max;
}
```

Заголовочный файл

maxnum.h > ...

```c
#ifndef MAXNUM_H
#define MAXNUM_H

int maxNum(int array[], int array_length);

#endif
```

Препроцессирование

Для препроцессирования необходимо выполнить следующие команды:

riscv64-unknown-elf-gcc -O1 -E main.c -o main.i
riscv64-unknown-elf-gcc -O1 -E maxnum.c -o maxnum.i

Соответственно результаты препроцессирования будут находится в файлах main.i и maxnum.i:

main.i:

main.i > ...

```
# 1 "main.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "main.c"

...

# 2 "main.c" 2
# 1 "maxnum.h" 1



# 4 "maxnum.h"
int maxNum(int array[], int array_length);
# 3 "main.c" 2

int main(void) {
    int array[] = {10, 5, 3, 2, 12, 69, 35, 69, 70, 11};
    int array_length = sizeof(array)/sizeof(array[0]);
    printf("%d", maxNum(array, array_length));
    return 0;
    }

```

maxnum.i:

maxnum.i > ...

```
# 1 "maxnum.c"
# 1 "<built-in>"
# 1 "<command-line>"
# 1 "maxnum.c"

...

# 2 "maxnum.c" 2
# 1 "maxnum.h" 1



# 4 "maxnum.h"
int maxNum(int array[], int array_length);
# 3 "maxnum.c" 2

int maxNum(int array[], int array_length) {
    int max = array[0];
    for (int i = 0; i < array_length; i++) {
        if (array[i] > max)
        {
            max = array[i];
        }
    }
    return max;
```

Компиляция

Для компилирования необходимо выполнить следующие команды:

riscv64-unknown-elf-gcc -O1 -S main.i -o main.s
riscv64-unknown-elf-gcc -O1 -S maxnum.i -o maxnum.s

Результат компилирования будет находится в файлах main.s и maxnum.s:

main.s:

main.s

```
    .file   "main.c"
    .option nopic
    .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .align  1
    .globl  main
    .type   main, @function
main:
    addi    sp,sp,-64
    sd  ra,56(sp)
    lui a5,%hi(.LANCHOR0)
    addi    a5,a5,%lo(.LANCHOR0)
    ld  a1,0(a5)
    ld  a2,8(a5)
    ld  a3,16(a5)
    ld  a4,24(a5)
    ld  a5,32(a5)
    sd  a1,8(sp)
    sd  a2,16(sp)
    sd  a3,24(sp)
    sd  a4,32(sp)
    sd  a5,40(sp)
    li  a1,10
    addi    a0,sp,8
    call    maxNum
    mv  a1,a0
    lui a0,%hi(.LC1)
    addi    a0,a0,%lo(.LC1)
    call    printf
    li  a0,0
    ld  ra,56(sp)
    addi    sp,sp,64
    jr  ra
    .size   main, .-main
    .section    .rodata
    .align  3
    .set    .LANCHOR0,. + 0
.LC0:
    .word   10
    .word   5
    .word   3
    .word   2
    .word   12
    .word   69
    .word   35
    .word   69
    .word   70
    .word   11
    .section    .rodata.str1.8,"aMS",@progbits,1
    .align  3
.LC1:
    .string "%d"
    .ident  "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

maxnum.s:

```
    .file   "maxnum.c"
    .option nopic
    .attribute arch, "rv64i2p0_m2p0_a2p0_f2p0_d2p0_c2p0"
    .attribute unaligned_access, 0
    .attribute stack_align, 16
    .text
    .align  1
    .globl  maxNum
    .type   maxNum, @function
maxNum:
    lw  a3,0(a0)
    ble a1,zero,.L2
    mv  a5,a0
    addiw   a1,a1,-1
    slli    a1,a1,32
    srli    a1,a1,32
    slli    a1,a1,2
    addi    a0,a0,4
    add a0,a1,a0
    j   .L4
.L3:
    sext.w  a3,a4
    addi    a5,a5,4
    beq a5,a0,.L2
.L4:
    lw  a4,0(a5)
    sext.w  a2,a4
    bge a2,a3,.L3
    mv  a4,a3
    j   .L3
.L2:
    mv  a0,a3
    ret
    .size   maxNum, .-maxNum
    .ident  "GCC: (SiFive GCC 8.3.0-2020.04.1) 8.3.0"
```

Ассемблирование

Для ассемблирования необходимо выполнить следующие команды:

riscv64-unknown-elf-gcc -v -c main.s -o main.o
riscv64-unknown-elf-gcc -v -c maxnum.s -o maxnum.o

В результате ассемблирования мы получаем объектные файлы, которые не являются текстовыми — используем утилиту objdump, чтобы получить содержимое бинарных файлов в текстовом виде. Используем следующую команду для отображения заголовков секций файлов:

riscv64-unknown-elf-objdump -h main.o

А для получения таблицы символов используем команду:

riscv64-unknown-elf-objdump -t main.o

Заголовки секций main.o:

```
Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000032  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000072  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000072  2**0
                  ALLOC
  3 .comment      00000029  0000000000000000  0000000000000000  00000072  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 00000035  0000000000000000  0000000000000000  0000009b  2**0
                  CONTENTS, READONLY
```

Таблица символов main.o:

```
main.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 maxnum.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
000000000000002e l       .text  0000000000000000 .L2
0000000000000020 l       .text  0000000000000000 .L4
0000000000000016 l       .text  0000000000000000 .L3
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  0000000000000032 maxNum
```

Для получения таблицы перемещений выполним команду:

riscv64-unknown-elf-objdump -d -M no-aliases -r main.o

Таблица перемещений main.o:

```
maxnum.o:     file format elf64-littleriscv


Disassembly of section .text:

0000000000000000 <maxNum>:
   0:   4114                    c.lw    a3,0(a0)
   2:   02b05663                bge     zero,a1,2e <.L2>
                        2: R_RISCV_BRANCH       .L2
   6:   87aa                    c.mv    a5,a0
   8:   35fd                    c.addiw a1,-1
   a:   1582                    c.slli  a1,0x20
   c:   9181                    c.srli  a1,0x20
   e:   058a                    c.slli  a1,0x2
  10:   0511                    c.addi  a0,4
  12:   952e                    c.add   a0,a1
  14:   a031                    c.j     20 <.L4>
                        14: R_RISCV_RVC_JUMP    .L4

0000000000000016 <.L3>:
  16:   0007069b                addiw   a3,a4,0
  1a:   0791                    c.addi  a5,4
  1c:   00a78963                beq     a5,a0,2e <.L2>
                        1c: R_RISCV_BRANCH      .L2

0000000000000020 <.L4>:
  20:   4398                    c.lw    a4,0(a5)
  22:   0007061b                addiw   a2,a4,0
  26:   fed658e3                bge     a2,a3,16 <.L3>
                        26: R_RISCV_BRANCH      .L3
  2a:   8736                    c.mv    a4,a3
  2c:   b7ed                    c.j     16 <.L3>
                        2c: R_RISCV_RVC_JUMP    .L3

000000000000002e <.L2>:
  2e:   8536                    c.mv    a0,a3
  30:   8082                    c.jr    ra
```

Аналогичные команды выполним для maxnum.o.

Заголовки секций maxnum.o:

```
maxnum.o:     file format elf64-littleriscv

Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .text         00000032  0000000000000000  0000000000000000  00000040  2**1
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  0000000000000000  0000000000000000  00000072  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  0000000000000000  0000000000000000  00000072  2**0
                  ALLOC
  3 .comment      00000029  0000000000000000  0000000000000000  00000072  2**0
                  CONTENTS, READONLY
  4 .riscv.attributes 00000035  0000000000000000  0000000000000000  0000009b  2**0
                  CONTENTS, READONLY
```

Таблица символов maxnum.o:

```
maxnum.o:     file format elf64-littleriscv

SYMBOL TABLE:
0000000000000000 l    df *ABS*  0000000000000000 maxnum.c
0000000000000000 l    d  .text  0000000000000000 .text
0000000000000000 l    d  .data  0000000000000000 .data
0000000000000000 l    d  .bss   0000000000000000 .bss
000000000000002e l       .text  0000000000000000 .L2
0000000000000020 l       .text  0000000000000000 .L4
0000000000000016 l       .text  0000000000000000 .L3
0000000000000000 l    d  .comment       0000000000000000 .comment
0000000000000000 l    d  .riscv.attributes      0000000000000000 .riscv.attributes
0000000000000000 g     F .text  0000000000000032 maxNum
```

Таблица перемещений maxnum.o:

```
Disassembly of section .text:

0000000000000000 <maxNum>:
   0:   4114                    c.lw    a3,0(a0)
   2:   02b05663                bge     zero,a1,2e <.L2>
                        2: R_RISCV_BRANCH       .L2
   6:   87aa                    c.mv    a5,a0
   8:   35fd                    c.addiw a1,-1
   a:   1582                    c.slli  a1,0x20
   c:   9181                    c.srli  a1,0x20
   e:   058a                    c.slli  a1,0x2
  10:   0511                    c.addi  a0,4
  12:   952e                    c.add   a0,a1
  14:   a031                    c.j     20 <.L4>
                        14: R_RISCV_RVC_JUMP    .L4

0000000000000016 <.L3>:
  16:   0007069b                addiw   a3,a4,0
  1a:   0791                    c.addi  a5,4
  1c:   00a78963                beq     a5,a0,2e <.L2>
                        1c: R_RISCV_BRANCH      .L2

0000000000000020 <.L4>:
  20:   4398                    c.lw    a4,0(a5)
  22:   0007061b                addiw   a2,a4,0
  26:   fed658e3                bge     a2,a3,16 <.L3>
                        26: R_RISCV_BRANCH      .L3
  2a:   8736                    c.mv    a4,a3
  2c:   b7ed                    c.j     16 <.L3>
                        2c: R_RISCV_RVC_JUMP    .L3

000000000000002e <.L2>:
  2e:   8536                    c.mv    a0,a3
  30:   8082                    c.jr    ra
```

Компоновка

Для компоновки необходимо выполнить следующую команду:

riscv64-unknown-elf-gcc -v main.o maxnum.o

При помощи следующей команды получаем фрагмент исполняемого файла a.out:

riscv64-unknown-elf-objdump -j .text -d -M no-aliases a.out > a.ds

```
0000000000010158 <main>:
   10158:   7139                c.addi16sp  sp,-64
   1015a:   fc06                c.sdsp   ra,56(sp)
   1015c:   67f1                c.lui a5,0x1c
   1015e:   0e078793            addi  a5,a5,224 # 1c0e0 <__clzdi2+0x36>
   10162:   638c                c.ld  a1,0(a5)
   10164:   6790                c.ld  a2,8(a5)
   10166:   6b94                c.ld  a3,16(a5)
   10168:   6f98                c.ld  a4,24(a5)
   1016a:   739c                c.ld  a5,32(a5)
   1016c:   e42e                c.sdsp   a1,8(sp)
   1016e:   e832                c.sdsp   a2,16(sp)
   10170:   ec36                c.sdsp   a3,24(sp)
   10172:   f03a                c.sdsp   a4,32(sp)
   10174:   f43e                c.sdsp   a5,40(sp)
   10176:   45a9                c.li  a1,10
   10178:   0028                c.addi4spn  a0,sp,8
   1017a:   018000ef            jal   ra,10192 <maxNum>
   1017e:   85aa                c.mv  a1,a0
   10180:   6571                c.lui a0,0x1c
   10182:   10850513            addi  a0,a0,264 # 1c108 <__clzdi2+0x5e>
   10186:   1da000ef            jal   ra,10360 <printf>
   1018a:   4501                c.li  a0,0
   1018c:   70e2                c.ldsp   ra,56(sp)
   1018e:   6121                c.addi16sp  sp,64
   10190:   8082                c.jr  ra

0000000000010192 <maxNum>:
   10192:   4114                c.lw  a3,0(a0)
   10194:   02b05663            bge   zero,a1,101c0 <maxNum+0x2e>
   10198:   87aa                c.mv  a5,a0
   1019a:   35fd                c.addiw  a1,-1
   1019c:   1582                c.slli   a1,0x20
   1019e:   9181                c.srli   a1,0x20
   101a0:   058a                c.slli   a1,0x2
   101a2:   0511                c.addi   a0,4
   101a4:   952e                c.add a0,a1
   101a6:   a031                c.j   101b2 <maxNum+0x20>
   101a8:   0007069b            addiw a3,a4,0
   101ac:   0791                c.addi   a5,4
   101ae:   00a78963            beq   a5,a0,101c0 <maxNum+0x2e>
   101b2:   4398                c.lw  a4,0(a5)
   101b4:   0007061b            addiw a2,a4,0
   101b8:   fed658e3            bge   a2,a3,101a8 <maxNum+0x16>
   101bc:   8736                c.mv  a4,a3
   101be:   b7ed                c.j   101a8 <maxNum+0x16>
   101c0:   8536                c.mv  a0,a3
   101c2:   8082                c.jr  ra
```

Создание статической библиотеки и Makefile

Для создания статической библиотеки необходимо получить объектные файлы всех используемых программ. Используем следующие команды:
riscv64-unknown-elf-gcc -c main.c -o main.o
riscv64-unknown-elf-gcc -c maxnum.c -o maxnum.o

Далее объединяем объектные файлы в библиотеку:
riscv64-unknown-elf-ar -rsc bsortlib.a maxnum.o

После получения библиотеки соберём исполняемый файл программы:
riscv64-unknown-elf-gcc -O1 --save-temps main.c bsortlib.a

Посмотрим таблицу символов исполняемого файла:
riscv64-unknown-elf-objdump -t a.out

В результате видим, что исполняемый файл содержит необходимые символы:

```
00000000000102d0 g     F .text  00000000000000aa memset
0000000000010158 g     F .text  000000000000003a main
000000000001e178 g     O .sbss  0000000000000008 __malloc_max_total_mem
000000000001a9f0 g     F .text  000000000000000c __swbuf
000000000001618c g     F .text  0000000000000008 __sclose
00000000000018e8e g     F .text  000000000000000a fclose
0000000000014d3e g     F .text  0000000000000660 _malloc_r
000000000001aaa0 g     F .text  0000000000000024 __ascii_wctomb
0000000000012a48 g     F .text  000000000000008a _fwalk
0000000000019420 g     F .text  000000000000000a _mbtowc_r
0000000000012706 g     F .text  00000000000000d8 _malloc_trim_r
0000000000019b84 g     F .text  00000000000000ea strcmp
00000000000018c3c g     F .text  0000000000000010 vfiprintf
000000000001aeee g     F .text  0000000000000606 .hidden __multf3
000000000001606e g     F .text  000000000000004c sprintf
000000000001cd40 g     O .rodata        0000000000000100 .hidden __clz_tab
000000000001e190 g     O .sbss  0000000000000008 _PathLocale
0000000000010202 g     F .text  000000000000000c atexit
00000000000018ce0 g     F .text  0000000000000040 _write_r
00000000000193d4 g     F .text  000000000000000c setlocale
000000000001e160 g     O .sdata 0000000000000008 _impure_ptr
00000000000122a0 g     F .text  0000000000000196 __sflush_r
000000000001ad92 g     F .text  00000000000000ae .hidden __gttf2
0000000000019dcc g     F .text  0000000000000b26 _svfiprintf_r
000000000001942a g     F .text  0000000000000040 __ascii_mbtowc
000000000001b4f4 g     F .text  000000000000081a .hidden __subtf3
0000000000015bf0 g     F .text  0000000000000060 __ulp
00000000000126f6 g     F .text  0000000000000010 __fp_unlock_all
0000000000014bfa g     F .text  0000000000000006 localeconv
0000000000014c00 g     F .text  0000000000000082 __swhatbuf_r
000000000001d020 g       .data  0000000000000000 __DATA_BEGIN__
000000000001ac62 g     F .text  0000000000000032 _write
000000000001e178 g       .sdata 0000000000000000 _edata
000000000001e208 g       .bss   0000000000000000 _end
00000000000018e98 g     F .text  00000000000000c6 __fputwc
00000000000160f6 g     F .text  0000000000000054 __swrite
000000000001e170 g     O .sdata 0000000000000008 __malloc_trim_threshold
000000000001020e g     F .text  0000000000000020 exit
0000000000017fc8 g     F .text  0000000000000c74 _vfiprintf_r
0000000000012ad2 g     F .text  0000000000000092 _fwalk_reent
0000000000015aa8 g     F .text  0000000000000148 __mdiff
00000000000126e0 g     F .text  0000000000000002 __sfp_lock_release
0000000000013cf0 g     F .text  0000000000000ec8 _ldtoa_r
000000000001cc00 g     O .rodata        0000000000000101 _ctype_
000000000001abc6 g     F .text  0000000000000032 _read
000000000001aafa g     F .text  000000000000002c _exit
0000000000014c82 g     F .text  00000000000000bc __smakebuf_r
0000000000016238 g     F .text  0000000000000098 strlen
0000000000017fba g     F .text  000000000000000e __sprint_r
0000000000010192 g     F .text  0000000000000070 maxNum
000000000001aa96 g     F .text  000000000000000a _wctomb_r
```

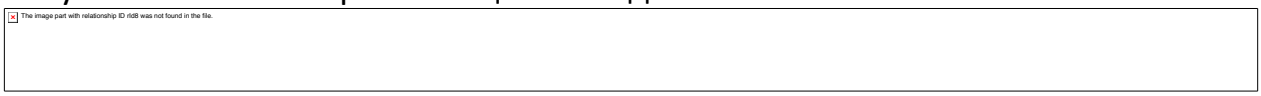
Важно отметить, что при компоновке с использование библиотеки компоновщик сам выбрал необходимые объектные файлы.

Создадим Makefile для автоматической сборки проекта, он будет выполнять следующие действия:

1. Получение объектного файла maxnum.o из maxnum.c
2. Архивация объектного файла maxnum.o и создание статической библиотеки bsortlib.a
3. Компоновка объектного файла main.c и библиотеки bsortlib.a для получения исполняемого файла a.out

Makefile

Запускаем Makefile при помощи команды make:



На выходе также получаем исполняемый файл. Если просмотреть таблицу символов, то видно, что содержимое файла точно такое же:

```
0000000000010158 g     F .text  000000000000003a main
000000000001e178 g     O .sbss  0000000000000008 __malloc_max_total_mem
000000000001a9f0 g     F .text  000000000000000c __swbuf
000000000001618c g     F .text  0000000000000008 __sclose
0000000000018e8e g     F .text  000000000000000a fclose
0000000000014d3e g     F .text  0000000000000660 _malloc_r
000000000001aaa0 g     F .text  0000000000000024 __ascii_wctomb
0000000000012a48 g     F .text  000000000000008a _fwalk
0000000000019420 g     F .text  000000000000000a _mbtowc_r
0000000000012706 g     F .text  00000000000000d8 _malloc_trim_r
0000000000019b84 g     F .text  00000000000000ea strcmp
0000000000018c3c g     F .text  0000000000000010 vfiprintf
000000000001aeee g     F .text  0000000000000606 .hidden __multf3
000000000001606e g     F .text  000000000000004c sprintf
000000000001cd40 g     O .rodata        0000000000000100 .hidden __clz_tab
000000000001e190 g     O .sbss  0000000000000008 _PathLocale
0000000000010202 g     F .text  000000000000000c atexit
0000000000018ce0 g     F .text  0000000000000040 _write_r
00000000000193d4 g     F .text  000000000000000c setlocale
000000000001e160 g     O .sdata 0000000000000008 _impure_ptr
00000000000122a0 g     F .text  0000000000000196 __sflush_r
000000000001ad92 g     F .text  00000000000000ae .hidden __gttf2
0000000000019dcc g     F .text  0000000000000b26 _svfiprintf_r
000000000001942a g     F .text  0000000000000040 __ascii_mbtowc
000000000001b4f4 g     F .text  000000000000081a .hidden __subtf3
0000000000015bf0 g     F .text  0000000000000060 __ulp
00000000000126f6 g     F .text  0000000000000010 __fp_unlock_all
0000000000014bfa g     F .text  0000000000000006 localeconv
0000000000014c00 g     F .text  0000000000000082 __swhatbuf_r
000000000001d020 g       .data  0000000000000000 __DATA_BEGIN__
000000000001ac62 g     F .text  0000000000000032 _write
000000000001e178 g       .sdata 0000000000000000 _edata
000000000001e208 g       .bss   0000000000000000 _end
0000000000018e98 g     F .text  00000000000000c6 __fputwc
00000000000160f6 g     F .text  0000000000000054 __swrite
000000000001e170 g     O .sdata 0000000000000008 __malloc_trim_threshold
000000000001020e g     F .text  0000000000000020 exit
0000000000017fc8 g     F .text  0000000000000c74 _vfiprintf_r
0000000000012ad2 g     F .text  0000000000000092 _fwalk_reent
0000000000015aa8 g     F .text  0000000000000148 __mdiff
00000000000126e0 g     F .text  0000000000000002 __sfp_lock_release
0000000000013cf0 g     F .text  0000000000000ec8 _ldtoa_r
000000000001cc00 g     O .rodata        0000000000000101 _ctype_
000000000001abc6 g     F .text  0000000000000032 _read
000000000001aafa g     F .text  000000000000002c _exit
0000000000014c82 g     F .text  00000000000000bc __smakebuf_r
0000000000016238 g     F .text  0000000000000098 strlen
0000000000017fba g     F .text  000000000000000e __sprint_r
0000000000010192 g     F .text  0000000000000070 maxNum
```

Вывод

Я реализовал программу для нахождения максимального элемента на языке C.

Проделал пошаговую сборку программы, а именно препроцессирование, компиляция, ассемблирование и компоновка.

Создал статическую библиотеку и автоматизировал сборку программы с помощью Makefile.