Отчет по лабораторной работе № 10 по курсу “Фундаментальная информатика”

Студент группы М80-101Б-22, Бычков Артур Сергеевич, № по списку 2

Контакты email: bychkovarthur@gmail.com

Работа выполнена: «12» ноября 2022г.

Преподаватель: каф. 806 Крылов Сергей Сергеевич

Входной контроль знаний с оценкой _____

Отчет сдан « » _____202 __ г., итоговая оценка _____

Подпись преподавателя _____

1. **Тема:** Отладчик системы программирования ОС Unix
2. **Цель работы:** Научиться отлаживать простейшие программы, написанные на языке Си

3. **Задание** (вариант №)**:** Нахождение корней квадратного уравнения (возможно ошибка деления на 0, если коэффициент a перед старшим членом равен нулю)

4. **Оборудование**:

Оборудование ПЭВМ студента, если использовалось:

Процессор **AMD Ryzen 5 5500U 2.10 GHz, 6 ядер** с ОП **8192 Мб**, ТТН **512000 Мб**. Мониторы **Lenovo.**

5. **Программное обеспечение:**

Программное обеспечение ЭВМ студента, если использовалось:

Операционная система семейства **Linux**, наименование **Ubuntu** версия **20.04.5,** интерпретатор команд bash версия 5.0.17(1).

Система программирования CLion версия 2021.1.3

Редактор текстов nano версия 6.2

Утилиты операционной системы WinRar, Microsoft Word.

Прикладные системы и программы Ubuntu wsl, Clion, Google Chrome

Местонахождение и имена файлов программ и данных на домашнем компьютере /home/artur

6. Идея, метод, алгоритм решения задачи (в формах: словесной, псевдокода, графической [блок-схема, диаграмма, рисунок, таблица] или формальные спецификации с пред- и постусловиями)

1. Скомпилировать программу при помощи ключа компилятора -g.
2. Введем данные, которые приводят к ошибке. Проделаем действия по отладке программы.

Основные команды отладчика (подчеркнуты минимальные сокращения)

Команда **gdb**	Описание команды
help [раздел]	Подсказка по разделу отладчика. Без параметров выводит список разделов.
list [имя фукнции/файла:] [номер строки]	Распечатка текста функции/процедуры/файла или всей программы, начиная с указанной строки. По умолчанию распечатываются 10 строк программы. Распечатываемый файл становится текущим файлом исходного текста отлаживаемой программы.
break [номер строки/имя функции]	Задание точки остановки на строке/функции текущего исходного файла программы
run [параметры]	Запуск программы на выполнение. Могут указываться необязательные параметры командной строки и операции перенаправления ввода-вывода. gdb запоминает параметры и подставляет их для дальнейших вызовах run.
set args [параметры]	Предварительная установка параметров командной строки.
show args	Вывод параметров командной строки.

print [выражение]	Печать значения выражения, которое может включать и переменные, и вызовы функций программы.
next [n]	Выполнение очередной строки программы при пошаговой трассировке (процедуры и функции не трассируются, а выполняются за один такт). Необязательный параметр n указывает число строк программы для выполнения. По умолчанию n = 1.
step [n]	Выполнение очередной строки программы (с трассировкой процедур и функций). Перед выполнением next/step программа должна быть запущена командой run.
set var [имя] = [выражение]	Присваивание значения переменной.
ptype [имя переменной]	Выводит тип переменной.
backtrace или **bt**	Распечатка содержимого стека вызовов.
continue	Продолжение выполнения программы после остановки.
quit	Выход из отладчика.

7. Сценарий выполнения работы [план работы, первоначальный текст программы в черновике (можно на отдельном листе) и тесты либо соображения по тестированию].

Тесты

№	Input (коэффициенты a, b, c уравнения)	Output
1	4 2 8	Уравнение не имеет решений
2	4 20 25	Корень уравнения x = -2
3	1 -6 -16	Корни уравнения x1 = -2, x2 = 8
4	0 1 2	Исключение в операции с плавающей точкой (стек памяти сброшен на диск)

1. Скомпилируем при помощи команды gcc и ключей -g и -lm программу lab10.c (gcc -g lab10.c -lm)
2. Запустим отладку с помощью команды gdb (gdb a.out)
3. Проделаем все действие по отладке программы.

8. Распечатка протокола (подклеить листинг окончательного варианта программы с тестовыми примерами, подписанный преподавателем).

```
artur@artur-VirtualBox:~$ gcc -g lab10.c -lm
artur@artur-VirtualBox:~$ gdb a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(gdb) help
List of classes of commands:

aliases -- Aliases of other commands.
breakpoints -- Making program stop at certain points.
data -- Examining data.
files -- Specifying and examining files.
internals -- Maintenance commands.
obscure -- Obscure features.
running -- Running the program.
stack -- Examining the stack.
status -- Status inquiries.
support -- Support facilities.
tracepoints -- Tracing of program execution without stopping the program.
user-defined -- User-defined commands.

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Type "apropos -v word" for full documentation of commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) help running
Running the program.

List of commands:

advance -- Continue the program up to the given location (same form as args for break command).
attach -- Attach to a process or file outside of GDB.
continue -- Continue program being debugged, after signal or breakpoint.
detach -- Detach a process or file previously attached.
detach checkpoint -- Detach from a checkpoint (experimental).
detach inferiors -- Detach from inferior ID (or list of IDS).
disconnect -- Disconnect from a target.
finish -- Execute until selected stack frame returns.
handle -- Specify how to handle signals.
inferior -- Use this command to switch between inferiors.
interrupt -- Interrupt the execution of the debugged program.
jump -- Continue program being debugged at specified line or address.
kill -- Kill execution of program being debugged.
kill inferiors -- Kill inferior ID (or list of IDs).
next -- Step program, proceeding through subroutine calls.
nexti -- Step one instruction, but proceed through subroutine calls.
queue-signal -- Queue a signal to be delivered to the current thread when it is resumed.
reverse-continue -- Continue program being debugged but run it in reverse.
reverse-finish -- Execute backward until just before selected stack frame is called.
reverse-next -- Step program backward, proceeding through subroutine calls.
```

```
reverse-nexti -- Step backward one instruction, but proceed through called subroutines.
reverse-step -- Step program backward until it reaches the beginning of another source line.
reverse-stepi -- Step backward exactly one instruction.
run -- Start debugged program.
signal -- Continue program with the specified signal.
start -- Start the debugged program stopping at the beginning of the main procedure.
starti -- Start the debugged program stopping at the first instruction.
step -- Step program until it reaches a different source line.
stepi -- Step one instruction exactly.
taas -- Apply a command to all threads (ignoring errors and empty output).
target -- Connect to a target machine or process.
target core -- Use a core file as a target.
target ctf -- (Use a CTF directory as a target.
target exec -- Use an executable file as a target.
target extended-remote -- Use a remote computer via a serial line, using a gdb-specific protocol.
target native -- Native process (started by the "run" command).
target record-btrace -- Collect control-flow trace and provide the execution history.
target record-core -- Log program while executing and replay execution from log.
target record-full -- Log program while executing and replay execution from log.
target remote -- Use a remote computer via a serial line, using a gdb-specific protocol.
target tfile -- Use a trace file as a target.
task -- Use this command to switch between Ada tasks.
tfaas -- Apply a command to all frames of all threads (ignoring errors and empty output).
--Type <RET> for more, q to quit, c to continue without paging--
thread -- Use this command to switch between threads.
thread apply -- Apply a command to a list of threads.
thread apply all -- Apply a command to all threads.
thread find -- Find threads that match a regular expression.
thread name -- Set the current thread's name.
until -- Execute until past the current line or past a LOCATION.

Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Type "apropos -v word" for full documentation of commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb)
(gdb) list
1    #include <stdio.h>
2    #include <math.h>
3
4
5    int main() {
6
7        int a = 0, b = 0, c = 0, x1 = 0, x2 = 0, D = 0;
8
9        printf("Введите коэффициенты a, b, c квадратного уравнения ax^2 + bx + c = 0\n");
10       scanf("%d%d%d", &a, &b, &c);
(gdb)
11
12       D = b * b - 4 * a * c;
13
14       if (D < 0) {
15       printf("Уравнение не имеет решений\n");
16       } else {
17
18       if (D == 0) {
19
20       x1 = (-b / (a * 2));
(gdb)
21       printf("Корень уравнения x = %d\n", x1);
22
23       } else {
24       x1 = (-b - sqrt(D)) / 2;
25       x2 = (-b + sqrt(D)) / 2;
26       x1 = x1 / a;
27       x2 = x2 / a;
28       printf("Корни уравнения x1 = %d, x2 = %d\n", x1, x2);
29
30       }
```

```
(gdb)
31          }
32  }
(gdb)
Line number 33 out of range; lab10.c has 32 lines.
(gdb) set args 123 321
(gdb) show args
Argument list to give program being debugged when it is started is "123 321".
(gdb) break 7
Breakpoint 1 at 0x11e4: file lab10.c, line 7.
(gdb) break 9
Breakpoint 2 at 0x120e: file lab10.c, line 9.
(gdb) break 14
Breakpoint 3 at 0x1254: file lab10.c, line 14.
(gdb) break 24
Breakpoint 4 at 0x129d: file lab10.c, line 24.
(gdb) break 25
Breakpoint 5 at 0x12d5: file lab10.c, line 25.
(gdb) break 26
Breakpoint 6 at 0x1305: file lab10.c, line 26.
(gdb) break 27
Breakpoint 7 at 0x1311: file lab10.c, line 27.
(gdb) break 29
Breakpoint 8 at 0x133b: file lab10.c, line 32.
(gdb) run
Starting program: /home/artur/a.out 123 321

Breakpoint 1, main () at lab10.c:7
7           int a = 0, b = 0, c = 0, x1 = 0, x2 = 0, D = 0;
(gdb) bt
#0  main () at lab10.c:7
(gdb) print a
$1 = 0
(gdb) p b
$2 = 0
(gdb) p c
$3 = 1431654624
(gdb) p x1
$4 = 21845
(gdb) p x2
$5 = -7952
(gdb) p D
$6 = 32767
(gdb) continue
Continuing.

Breakpoint 2, main () at lab10.c:9
9           printf("Введите коэффициенты a, b, c квадратного уравнения ax^2 + bx + c = 0\n");
(gdb) p a
$7 = 0
(gdb) p b
$8 = 0
(gdb) p c
$9 = 0
(gdb) p x1
$10 = 0
(gdb) p x2
$11 = 0
(gdb) p D
$12 = 0
(gdb) set var a = -100
(gdb) p a
$13 = -100
(gdb) c
Continuing.
Введите коэфициэнты a, b, c квадратного уравнения ax^2 + bx + c = 0
0 1 2

Breakpoint 3, main () at lab10.c:14
```

```
14          if (D < 0) {
(gdb) p a
$14 = 0
(gdb) p b
$15 = 1
(gdb) p c
$16 = 2
(gdb) p D
$17 = 1
(gdb) c
Continuing.

Breakpoint 4, main () at lab10.c:24
24          x1 = (-b - sqrt(D)) / 2;
(gdb) p x1
$18 = 0
(gdb) c
Continuing.

Breakpoint 5, main () at lab10.c:25
25          x2 = (-b + sqrt(D)) / 2;
(gdb) p x2
$19 = 0
(gdb) p x1
$20 = -1
(gdb) c
Continuing.

Breakpoint 6, main () at lab10.c:26
26          x1 = x1 / a;
(gdb) p x1
$21 = -1
(gdb) p a
$22 = 0
(gdb) c
Continuing.

Program received signal SIGFPE, Arithmetic exception.
0x000055555555530c in main () at lab10.c:26
26          x1 = x1 / a;
(gdb) continue
Continuing.

Program terminated with signal SIGFPE, Arithmetic exception.
The program no longer exists.
(gdb) q
artur@artur-VirtualBox:~$
```

9. Дневник отладки должен содержать дату и время сеансов отладки и основные события (ошибки в сценарии и программе, нестандартные ситуации) и краткие комментарии к ним. В дневнике отладки приводятся сведения об использовании других ЭВМ, существенном участии преподавателя и других лиц в написании и отладке программы.

№	Лаб. или дом.	Дата	Время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы

Я научился отлаживать простейшие программы, написанные на языке Си.

Недочёты при выполнении задания могут быть устранены следующим образом: --

Подпись студента

_____