**Отчет по лабораторной работе №  3(8)**

по курсу **Фундаментальная информатика**

**Выполнила:** М8О-111Б-23  Тимофеева Ирина Александровна, № по списку  20

**Работа выполнена:** «15» ноября  2023 г.

        Преподаватель:  доцент каф. 806 Никулин Сергей Петрович

Отчет сдан «20» ноября 2023 г., итоговая оценка \_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Тема:**Системы программирования на языке C

**2. Цель работы:**Изучение конкретной системы программирования на Си и получение

навыков подготовки текстов и отладки программ.

**3. Оборудование**(лабораторное):

ЭВМ \_\_\_\_\_\_\_\_\_, процессор \_\_\_\_\_\_\_.  имя узла сети\_\_\_\_\_\_\_сОП\_\_\_\_\_ Мб, НМД \_\_\_\_ Мб. Терминал \_\_\_\_\_\_\_\_адрес \_\_\_\_\_\_\_\_. Принтер\_\_\_\_\_\_\_

Другие устройства

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Оборудование ПЭВМ студента, если использовалось:*

Процессор \_Intel Core i5 -12500H\_ с ОП    16   Гб  НМД \_\_\_512\_\_Гб. Монитор   1920x1080~144Hz        Другие устройства

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

1) Написать рабочий код программы (вычисление среднего арифметического набора чисел).

2) Скомпилировать и запустить рабочий код.

3) Допустить в программе синтаксическую ошибку.

4) Допустить ошибку неправильного вывода данных.

5) Воспользоваться отладчиком gdb.

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

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

**Рабочая версия программы:**

// C program to demonstrate

// average of array elements

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_LENGTH 10

#define NUMBERS\_AMOUNT 1000000

// Function prints array

void printArray(int\* array, int length) {

int i;

for (i = 0; i < length; i++)

printf("%d ", array[i]);

printf("\n");

}

// Function that return average

// of given array.

double average(int a[], int n)

{

// Find the sum of array element

int sum = 0;

for (int i = 0; i < n; i++)

sum += a[i];

return (double)sum / n;

}

int\* arrayInit(int l) {

return calloc(l, sizeof(int));

}

int main()

{

// init/randomize

srand(time(NULL));

int a, i;

int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

// set random values of array items

for (i = 0; i < ARRAY\_LENGTH; i++)

\*(rnd\_array + i) = (int)rand() % 100;

// Display result

printf("Our array: ");

printArray(rnd\_array, ARRAY\_LENGTH);

// average(arr, n) function is return the

// average of the array.

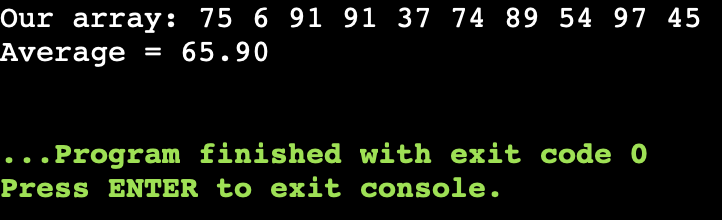
double avg = average(rnd\_array, ARRAY\_LENGTH);

// Display average of given array

printf("Average = %.2f\n", avg);

return 0;

}

**Результат:**

**Код c синтаксической ошибкой:**

// C program to demonstrate

// average of array elements

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_LENGTH 10

#define NUMBERS\_AMOUNT 1000000

// Function prints array

void printArray(int\* array, int length) {

int i;

for (i = 0; i < length; i++)

printf("%d ", array[i]);

printf("\n");

}

// Function that return average

// of given array.

double average(int a[], int n)

{

// Find the sum of array element

int sum = 0;

for (int i = 0; i < n; i++)

sum += a[i];

return (double)sum / n;

}

int\* arrayInit(int l) {

return calloc(l, sizeof(int));

}

int main()

{

// init/randomize

srand(time(NULL));

int a, i;

int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

// set random values of array items

for (i = 0; i < ARRAY\_LENGTH; i++)

\*(rnd\_array + i) = (int)rand() % 100;

// Display result

printf("Our array: ");

printArray(Rnd\_array, ARRAY\_LENGTH);

// average(arr, n) function is return the

// average of the array.

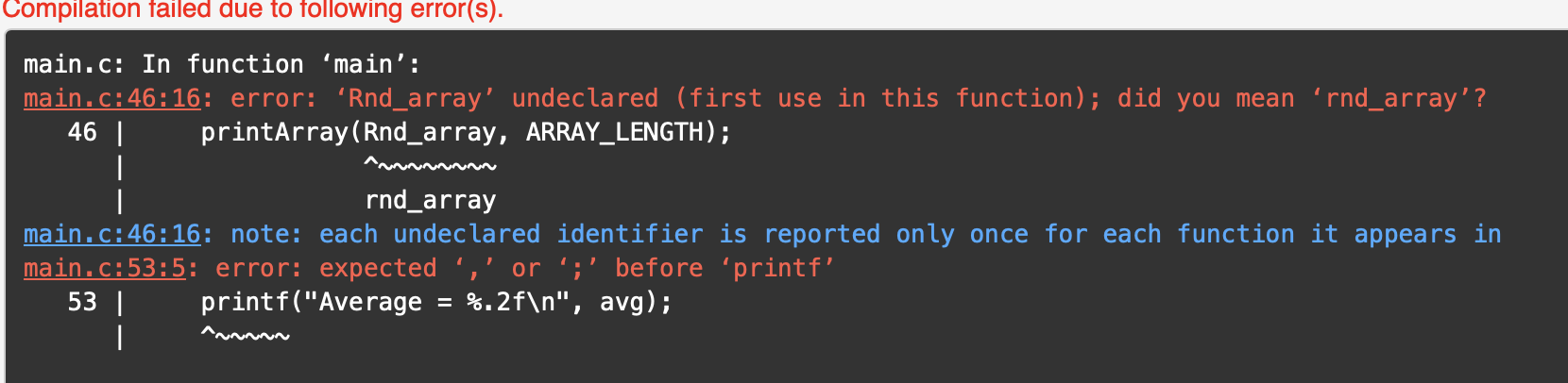
double avg = average(rnd\_array, ARRAY\_LENGTH)

// Display average of given array

printf("Average = %.2f\n", avg);

return 0;

}

**Результат:**

**Код с ошибкой:**

// C program to demonstrate

// average of array elements

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_LENGTH 10

#define NUMBERS\_AMOUNT 1000000

// Function prints array

void printArray(int\* array, int length) {

int i;

for (i = 0; i < length; i++)

printf("%d ", array[i]);

printf("\n");

}

// Function that return average

// of given array.

double average(int a[], int n)

{

// Find the sum of array element

int sum = 0;

for (int i = 0; i < n; i++)

sum += a[i];

return (double)sum / n;

}

int\* arrayInit(int l) {

return calloc(l, sizeof(int));

}

int main()

{

// init/randomize

srand(time(NULL));

int a, i;

int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

// set random values of array items

// add range to show memory allocation error

for (i = 0; i <= ARRAY\_LENGTH; i++)

\*(rnd\_array + i) = (int)rand() % 100;

// Display result

printf("Our array: ");

printArray(rnd\_array, ARRAY\_LENGTH);

// average(arr, n) function is return the

// average of the array.

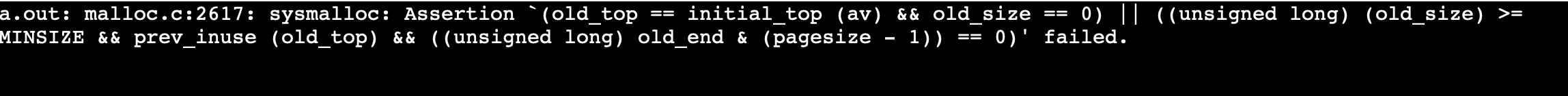
double avg = average(rnd\_array, ARRAY\_LENGTH);

// Display average of given array

printf("Average = %.2f\n", avg);

return 0;

}

**Результат:**

**gdb отладка:**

//Программа без ошибок

irina@Irina-VivoBook:~/Prog$ cat array.c

// C program to demonstrate

// average of array elements

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_LENGTH 10

#define NUMBERS\_AMOUNT 1000000

// Function prints array

void printArray(int\* array, int length) {

int i;

for (i = 0; i < length; i++)

printf("%d ", array[i]);

printf("\n");

}

// Function that return average

// of given array.

double average(int a[], int n)

{

// Find the sum of array element

int sum = 0;

for (int i = 0; i < n; i++)

sum += a[i];

return (double)sum / n;

}

int\* arrayInit(int l) {

return calloc(l, sizeof(int));

}

int main()

{

// init/randomize

srand(time(NULL));

int a, i;

int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

// set random values of array items

for (i = 0; i < ARRAY\_LENGTH; i++)

\*(rnd\_array + i) = (int)rand() % 100;

// Display result

printf("Our array: ");

printArray(rnd\_array, ARRAY\_LENGTH);

// average(arr, n) function is return the

// average of the array.

double avg = average(rnd\_array, ARRAY\_LENGTH);

// Display average of given array

printf("Average = %.2f\n", avg);

return 0;

}

irina@Irina-VivoBook:~/Prog$ gcc array.c

irina@Irina-VivoBook:~/Prog$ ./a.out

Our array: 2 83 95 20 18 19 62 44 83 99

Average = 52.50

//Программа с синтаксическими ошибками

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_LENGTH 10

#define NUMBERS\_AMOUNT 1000000

// Function prints array

void printArray(int\* array, int length) {

int i;

for (i = 0; i < length; i++)

printf("%d ", array[i]);

printf("\n");

}

// Function that return average

// of given array.

double average(int a[], int n)

{

// Find the sum of array element

int sum = 0;

for (int i = 0; i < n; i++)

sum += a[i];

return (double)sum / n;

}

int\* arrayInit(int l) {

return calloc(l, sizeof(int));

}

int main()

{

// init/randomize

srand(time(NULL));

int a, i;

int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

// set random values of array items

for (i = 0; i < ARRAY\_LENGTH; i++)

\*(rnd\_array + i) = (int)rand() % 100;

// Display result

printf("Our array: ");

printArray(Rnd\_array, ARRAY\_LENGTH);

// average(arr, n) function is return the

// average of the array.

double avg = average(rnd\_array, ARRAY\_LENGTH)

// Display average of given array

printf("Average = %.2f\n", avg);

return 0;

}

irina@Irina-VivoBook:~/Prog$ gcc array\_syntax.c -lm

array\_syntax.c: In function ‘main’:

array\_syntax.c:46:16: error: ‘Rnd\_array’ undeclared (first use in this function); did you mean ‘rnd\_array’?

46 | printArray(Rnd\_array, ARRAY\_LENGTH);

| ^~~~~~~~~

| rnd\_array

array\_syntax.c:46:16: note: each undeclared identifier is reported only once for each function it appears in

array\_syntax.c:53:5: error: expected ‘,’ or ‘;’ before ‘printf’

53 | printf("Average = %.2f\n", avg);

// Добавили ошибку обращения к не объявленной ячейке массива

irina@Irina-VivoBook:~/Prog/Prog\_C$ cat array.c

// C program to demonstrate

// average of array elements

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define ARRAY\_LENGTH 10

#define NUMBERS\_AMOUNT 1000000

// Function prints array

void printArray(int\* array, int length) {

int i;

for (i = 0; i < length; i++)

printf("%d ", array[i]);

printf("\n");

}

// Function that return average

// of given array.

double average(int a[], int n)

{

// Find the sum of array element

int sum = 0;

for (int i = 0; i < n; i++)

sum += a[i];

return (double)sum / n;

}

int\* arrayInit(int l) {

return calloc(l, sizeof(int));

}

int main()

{

// init/randomize

srand(time(NULL));

int a, i;

int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

// set random values of array items

for (i = 0; i < ARRAY\_LENGTH; i++)

\*(rnd\_array + i) = (int)rand() % 100;

// Display result

printf("Our array: ");

printArray(rnd\_array, ARRAY\_LENGTH);

// average(arr, n) function is return the

// average of the array.

double avg = average(rnd\_array, ARRAY\_LENGTH);

// Display average of given array

printf("Average = %.2f\n", avg);

return 0;

}

irina@Irina-VivoBook:~/Prog/Prog\_C$ gcc array.c

irina@Irina-VivoBook:~/Prog/Prog\_C$ gdb ./a.out

GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1

Copyright (C) 2022 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:

<https://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) list 1

1 // C program to demonstrate

2 // average of array elements

3 #include <stdio.h>

4 #include <stdlib.h>

5 #include <time.h>

6 #define ARRAY\_LENGTH 10

7 #define NUMBERS\_AMOUNT 1000000

8

9 // Function prints array

10 void printArray(int\* array, int length) {

(gdb)

11 int i;

12 for (i = 0; i < length; i++)

13 printf("%d ", array[i]);

14 printf("\n");

15 }

16

17 // Function that return average

18 // of given array.

19 double average(int a[], int n)

20 {

(gdb)

21 // Find the sum of array element

22 int sum = 0;

23 for (int i = 0; i < n; i++)

24 sum += a[i];

25

26 return (double)sum / n;

27 }

28

29 int\* arrayInit(int l) {

30 return calloc(l, sizeof(int));

(gdb)

31 }

32

33 int main()

34 {

35 // init/randomize

36 srand(time(NULL));

37 int a, i;

38 int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

39

40 // set random values of array items

(gdb)

41 for (i = 0; i < ARRAY\_LENGTH; i++)

42 \*(rnd\_array + i) = (int)rand() % 100;

43

44 // Display result

45 printf("Our array: ");

46 printArray(rnd\_array, ARRAY\_LENGTH);

47

48 // average(arr, n) function is return the

49 // average of the array.

50 double avg = average(rnd\_array, ARRAY\_LENGTH);

(gdb)

51

52 // Display average of given array

53 printf("Average = %.2f\n", avg);

54

55 return 0;

56 }

(gdb)

(gdb) break main

Breakpoint 1 at 0x12df: file array.c, line 36.

(gdb) b 41

Breakpoint 2 at 0x12fe: file array.c, line 41.

(gdb) info break

Num Type Disp Enb Address What

1 breakpoint keep y 0x00005555555552df in main at array.c:36

breakpoint already hit 1 time

2 breakpoint keep y 0x00005555555552fe in main at array.c:41

breakpoint already hit 1 time

(gdb) run

The program being debugged has been started already.

Start it from the beginning? (y or n) y

Starting program: /home/irina/Prog/Prog\_C/a.out

[Thread debugging using libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

Breakpoint 1, main () at array.c:36

36 srand(time(NULL));

1: rnd\_array = (int \*) 0x0

2: \*(rnd\_array + i) = <error: Cannot access memory at address 0xfffffffe00000000>

3: i = -2147483648

(gdb) n

38 int\* rnd\_array = arrayInit(ARRAY\_LENGTH);

1: rnd\_array = (int \*) 0x0

2: \*(rnd\_array + i) = <error: Cannot access memory at address 0xfffffffe00000000>

3: i = -2147483648

(gdb)

Breakpoint 2, main () at array.c:41

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = <error: Cannot access memory at address 0x5553555592a0>

3: i = -2147483648

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 0

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 26

3: i = 0

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 1

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 97

3: i = 1

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 2

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 41

3: i = 2

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 3

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 6

3: i = 3

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 4

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 91

3: i = 4

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 5

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 15

3: i = 5

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 6

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 46

3: i = 6

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 7

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 17

3: i = 7

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 8

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 1

3: i = 8

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 9

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 66

3: i = 9

(gdb)

42 \*(rnd\_array + i) = (int)rand() % 100;

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 134465

3: i = 10

(gdb)

41 for (i = 0; i <= ARRAY\_LENGTH; i++)

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 36

3: i = 10

(gdb)

45 printf("Our array: ");

1: rnd\_array = (int \*) 0x5555555592a0

2: \*(rnd\_array + i) = 0

3: i = 11

(gdb)

a.out: malloc.c:2617: sysmalloc: Assertion `(old\_top == initial\_top (av) && old\_size == 0) || ((unsigned long) (old\_size) >= MINSIZE && prev\_inuse (old\_top) && ((unsigned long) old\_end & (pagesize - 1)) == 0)' failed.

Program received signal SIGABRT, Aborted.

\_\_pthread\_kill\_implementation (no\_tid=0, signo=6, threadid=140737353779008) at ./nptl/pthread\_kill.c:44

44 ./nptl/pthread\_kill.c: Нет такого файла или каталога.

(gdb) c

Continuing.

Program terminated with signal SIGABRT, Aborted.

The program no longer exists.

(gdb) Quit

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.

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, fg, c -- 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, fin -- 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, j -- 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, n -- Step program, proceeding through subroutine calls.

nexti, ni -- 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, rc -- Continue program being debugged but run it in reverse.

reverse-finish -- Execute backward until just before selected stack frame is called.

reverse-next, rn -- Step program backward, proceeding through subroutine calls.

reverse-nexti, rni -- Step backward one instruction, but proceed through called subroutines.

reverse-step, rs -- Step program backward until it reaches the beginning of another source line.

reverse-stepi, rsi -- Step backward exactly one instruction.

run, r -- Start debugged program.

signal -- Continue program with the specified signal.

--Type <RET> for more, q to quit, c to continue without paging--

start -- Start the debugged program stopping at the beginning of the main procedure.

starti -- Start the debugged program stopping at the first instruction.

step, s -- Step program until it reaches a different source line.

stepi, si -- 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.

task apply -- Apply a command to a list of tasks.

task apply all -- Apply a command to all tasks in the current inferior.

tfaas -- Apply a command to all frames of all threads (ignoring errors and empty output).

thread, t -- 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, u -- 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 <RET> for more, q to quit, c to continue without paging--

Type "apropos -v word" for full documentation of commands related to "word".

Command name abbreviations are allowed if unambiguous.

**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**

В ходе выполнения работы, я научилась писать простейшие программы на языке программирования С, а так же научился работать с отладкой программы в gdb .

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_