

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра «Вычислительной техники»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
по курсу «Программирование»
на тему «Игра Сапер»

Выполнил студенты группы 23ВВВ4:

Костин Кирилл

Принял:

Голотенков Н.О.

от

03.06.24.

Пенза 2024

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет Вычислительной техники
Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. кафедрой ВТ



М.А. Митрохин

« ____ » _____ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Программирование

Студенту Костину Кириллу Алексеевичу Группа 23BVB4

Тема проекта: *Разработка программы сложной структуры методом нисходящего программирования.*

Исходные данные (технические требования) на проектирование

Разработать компьютерную игру «Сапер». В игре пользователь соревнуется с ПК. Размер минного поля не менее 30 на 30 клеток. Пользователь может помечать ячейки с минами и открывать пустые ячейки. Смежные пустые ячейки должны открываться автоматически. Пользователь проиграл, если была открыта ячейка с миной. Если открыты все пустые ячейки пользователь выиграл. 10 лучших результатов сохранять в файл и выводить их в конце каждой игры.

Обязательные требования к программе:

1. Многомодульность.
2. Использование сложных типов данных – структур, списков, файлов.
3. Режим работы видеосистемы – текстовый / графический.
4. Устройство ввода информации – клавиатура, мышь.

5. Пользовательский интерфейс должен быть построен на основе меню и панели инструментов.
6. Наличие заставки.
7. Операционная система MS Windows.
8. Язык программирования – Си и Ассемблер.
9. Среда разработки ПО – Microsoft Visual Studio.

Объем работы по курсу

1. Расчетная часть

Разработка программы.

2. Графическая часть

Схема данных, схема ресурсов системы, схема работы системы, иерархическая структура программы, схема взаимодействия программы.

3. Экспериментальная часть

Отладка программы.

Срок выполнения проекта по разделам

В соответствии с графиком.

Дата выдачи задания “ 09 ” 02. 2024

Дата защиты проекта “ ”

Руководитель Блохенко В.Н.О.

Задание получил “ 09 ” февраля 2024г.

Студент Костин Кирилл Алексеевич

Содержание

Содержание	4
Введение.....	5
1. Постановка задачи	6
2. Выбор решений	7
3. Отладка и тестирование.....	8
3.1. Установка точки останова и запуск отладчика	9
3.2. Настройка отображения	10
3.3. Переход по коду в отладчике с помощью пошаговых команд	13
4. Описание программы	14
5. Схема программы	16
6. Руководство пользователя	18
6.1. Вступительный диалог.....	18
6.2. Игровой процесс	21
Заключение.....	25
Приложение А	26
Листинг программы.....	26
Файл saper.c.....	26
Файл asm-x86_64.c	32
Файл gamefn.c	33
Файл term_tools.c	46
Приложение В	51

Введение

На сегодняшний день, почти все люди играют в игры. В ходе развития мощностей ЭВМ, также развивались и игры. От Pac-Man (1980) до актуальных Atomic Heart, Red Dead Redemption 2 и др. За столь долгую историю, игры эволюционировали как в графике, так и в других не менее важных аспектах, в следствии чего более старые игры, которые раньше считались популярными, припадают забвению. Одной из таких игр является игра «Сапер».

История игры «Сапер» берёт своё начало в 1980-х годах, когда она была создана как часть набора игр, поставляемых с операционной системой Microsoft Windows. Целью игры «Сапер» является расчистка игрового поля от мин, которые спрятаны под клетками. Игрок должен открывать клетки, не зная, содержит ли она мину или нет. Если игрок открывает клетку с миной, то он проигрывает игру. Если игрок успешно расчистит все клетки, не содержащие мин, то он выигрывает игру. Несмотря на свою простоту, игра «Сапер» требует от игрока хороших навыков логического мышления и внимательности.

1. Постановка задачи

Необходимо разработать игру с несколькими уровнями сложности и динамическим размером поля. Игра должна при старте запрашивать у пользователя размер поля в формате ХхУ и предупреждать, что размер меньше рекомендуемого или больше окна терминала. Также в конце каждой игры количество очков должно заноситься в файл.

Многомодульность программы. Программа должна быть поделена на логические модули. Это упростит поиск ошибок при отладке и тестировании программы, а также позволит легко расширять функционал программы. Использование сложных типов данных – массивов, структур, файлов. Это необходимо для более простой и интуитивной обработки данных в коде программы.

Режим работы видеосистемы – текстовый/графический. Для начала необходимо определиться с типом интерфейса и с элементами управления, затем необходимо изучить способы их реализации.

Устройство ввода-вывода – клавиатура и мышь. Необходимо различать и идентифицировать действия, произведенные с их помощью, это облегчит использование программы.

Интерфейс должен быть построен на основе меню. Это необходимо для создания интуитивно понятного интерфейса для пользователя.

Заставка необходима для того, чтобы пользователь, запустивший программу, смог получить достаточную информацию о ней.

2. Выбор решений

При запуске игры открывается приветственный экран. Далее запускается диалог с пользователем:

1. Указать размер поля.
2. (Ситуативно) уточнить, точно ли пользователь согласен с тем, что размер поля меньше рекомендуемого/больше окна терминала.
3. Указать сложность.

После этого диалога начинается сама игра:

1. Отрисовка игрового поля.
 - a. Отображение вспомогательной информации.
 - b. Отрисовка поля.
 - c. Отображение служебной информации.
2. Ожидания действия со стороны пользователя.
 - a. Если введено w/a/s/d h/j/k/l - передвинуть курсор в указанную сторону.
 - b. Если введено c/p - открыть все пустые соседние клетки (если начальная клетка была бомбой, вывести сообщение о поражении).
 - c. Если введено f - поставить/убрать флаг на еще закрытую клетку.
 - d. Если введено ` - открыть приглашение к вводу команды в игровую консоль.

- i. Если введено `drowb` - включить отрисовку бомб (упрощает отладку, пользователь может воспринимать как читы).
 - ii. Если введено `ndrowb` - выключить отрисовку бомб.
 - iii. Если введено `lead` - отобразить лидерборд.
 - e. Если введено `q` - запустить диалог выхода из игры.
 - i. Если введено `yes` — закрыть игру.
 - ii. Если введено `no` — закрыть диалог, вернуть пользователя в игру.
 - f. Если введено `r` - перерисовать поле.
 - g. Если введено `2` - выключить фоновую музыку.
 - h. Если введено `T` — перезапустить игру.
3. Если открыты все клетки вывести сообщение о победе.
- a. Отобразить результат пользователя
 - b. Сохранить результат пользователя в файл.
4. Если игра была завершена и не проиграна, запускается внешняя программ **cowsay** с сообщением

3. Отладка и тестирование

В качестве среды для написания кода была использована собственная сборка консольного тестового редактора **Neovim** с плагинами.

В качестве программы отладчика был использован **GDB** . Эта программа обладает всеми необходимыми функциями для отладки кода, написанного на языке Си, такие как:

1. Точка останова
2. Трассировка
3. Анализ содержимого переменных
4. Просмотр ассемблерного кода

В качестве компилятора был использован **clang**.

Разработка проводилась на операционной системе **Endeavor Os Linux**.

Связка этих программ и операционной системы позволяет вести быструю и эффективную разработку в терминальной среде UNIX.

3.1. Установка точки останова и запуск отладчика

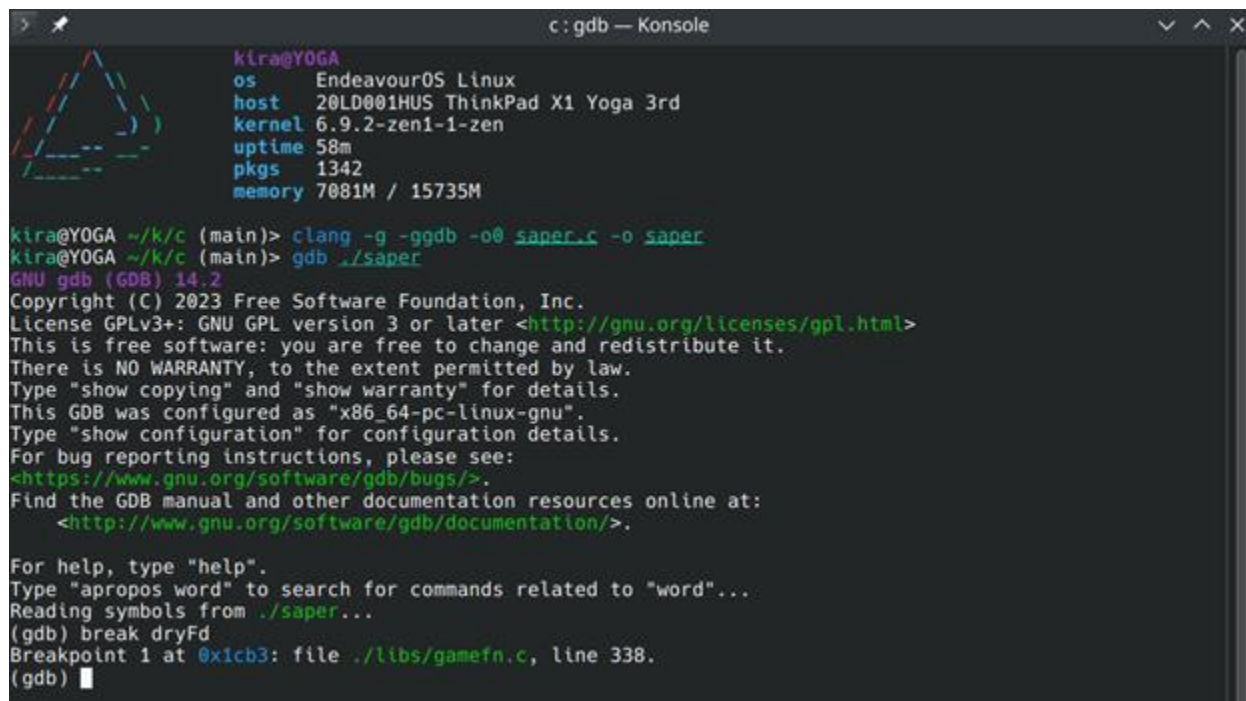
Точки останова полезны, если вам известны строка или раздел кода, которые вы хотите подробно изучить в среде выполнения. Дополнительные сведения о различных типах точек останова, которые можно задать, например, об условных точках останова и точках останова в функциях.

Для корректной отладки нужно скомпилировать приложение с отладочной информацией. Сделать это можно командой **clang -g -ggdb -o0 saper.c -o saper**. Далее нужно запустить программу в среде отладчика командой **gdb ./saper**.

После запуска среды можно указать точку останова следующими способами:

1. Указать название функции
2. Указать название файла и номер строки

Далее нужно запустить исполнение программы командой **run**

A screenshot of a terminal window titled 'c: gdb — Konsole'. The terminal shows a user 'kira@YOGA' on an 'EndeavourOS Linux' system. System information is displayed: host '20LD001HUS ThinkPad X1 Yoga 3rd', kernel '6.9.2-zen1-1-zen', uptime '58m', pkgs '1342', and memory '7081M / 15735M'. The user runs 'clang -g -ggdb -o0 saper.c -o saper' and then 'gdb ./saper'. The GDB version is 14.2. A copyright notice for the GNU Free Software Foundation is shown. The user sets a breakpoint at 'dryFd' in 'libs/gamefn.c' at line 338. The prompt '(gdb) ' is visible at the bottom.

```
c: gdb — Konsole
kira@YOGA
os      EndeavourOS Linux
host    20LD001HUS ThinkPad X1 Yoga 3rd
kernel  6.9.2-zen1-1-zen
uptime  58m
pkgs    1342
memory  7081M / 15735M

kira@YOGA ~/k/c (main)> clang -g -ggdb -o0 saper.c -o saper
kira@YOGA ~/k/c (main)> gdb ./saper
GNU gdb (GDB) 14.2
Copyright (C) 2023 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-pc-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 ./saper...
(gdb) break dryFd
Breakpoint 1 at 0x1cb3: file ./libs/gamefn.c, line 338.
(gdb) 
```

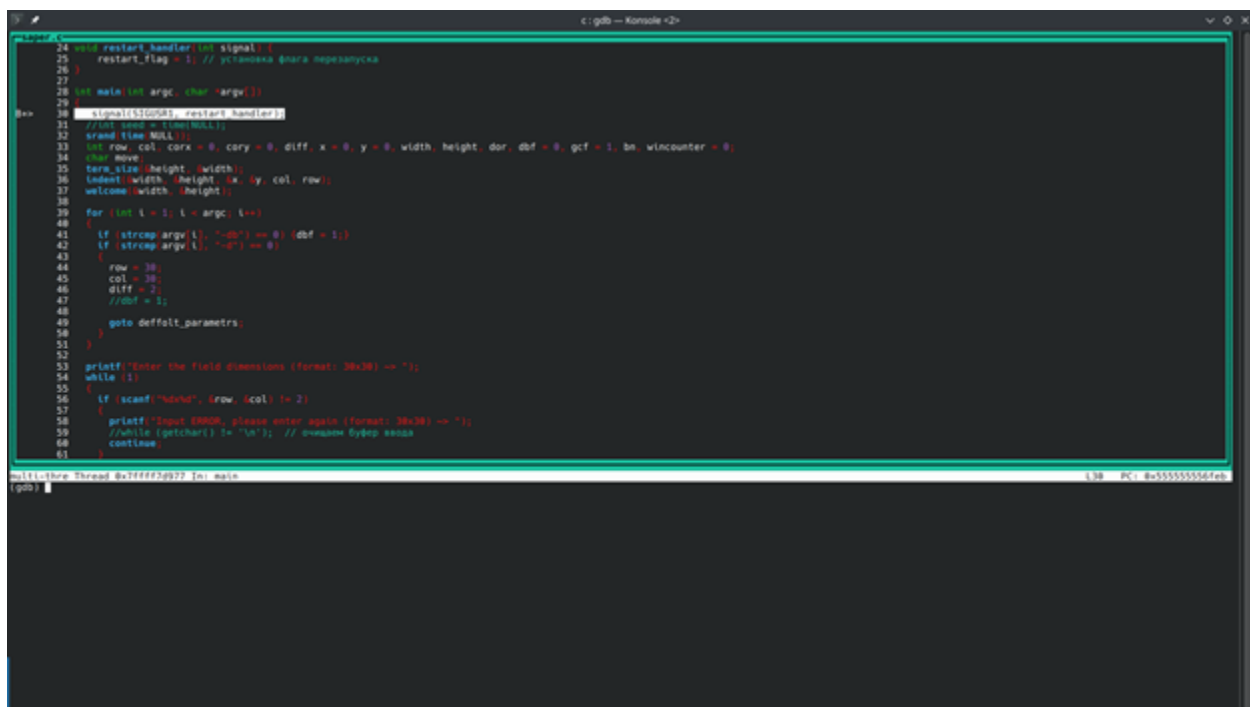
Рисунок 1 – Компилирование, установка точки останова

3.2. Настройка отображения

GDB предоставляет возможность настройки отображения информации, которая показывается во время отладки. Ниже приведены команды **GDB**, которые, были использованы в ходе отладки приложения:

1. **Команда layout.** Команда layout позволяет вам настроить расположение окон на экране, layout src (показать окно исходного кода), layout asm

(показать окно ассемблерного кода) и layout split (показать ассемблерный и исходный код одновременно).



```
24 void restart_handler(int signal) {
25     restart_flag = 1; // ychayakna flaga pereizanyuka
26 }
27
28 int main(int argc, char *argv[])
29 {
30     signal(SIGINT, restart_handler);
31     //rand_sseed = 1000000000;
32     srand(time(NULL));
33     int row, col, corx = 0, cory = 0, diff, x = 0, y = 0, width, height, dor, dbf = 0, gcf = 1, bn, wincounter = 0;
34     //for move
35     term_size(&height, &width);
36     initscr(&width, &height, &x, &y, &col, &row);
37     welcome(&width, &height);
38
39     for (int i = 1; i < argc; i++)
40     {
41         if (strcmp(argv[i], "-db") == 0) {dbf = 1;}
42         if (strcmp(argv[i], "-g") == 0)
43         {
44             row = 30;
45             col = 30;
46             diff = 2;
47             //dbf = 1;
48         }
49         goto default_params;
50     }
51
52     printf("Enter the field dimensions (format: 30x30) ==> ");
53     while (1)
54     {
55         if (scanf("%dxd%d", &row, &col) != 2)
56         {
57             printf("Input ERROR, please enter again (format: 30x30) ==> ");
58             //while (getchar() != '\n'); // очищаем буфер ввода
59             continue;
60         }
61     }
```

Рисунок 2 – Отображение исходного кода

```

0x55555556f08 <restart_handler>      push    %rbp
0x55555556f09 <restart_handler+1>      mov     %rbp,%rbp
0x55555556f0a <restart_handler+4>      mov     %eax,%eax
0x55555556f0b <restart_handler+7>      movl    $0x1,%eax
0x55555556f0c <restart_handler+10>     pop     %rbp
0x55555556f0d <restart_handler+13>     ret
0x55555556f0e <restart_handler+16>     data16  data16 cs nops 0x0(0x0,0x0,1)
0x55555556f0f <restart_handler+19>     push    %rbp
0x55555556f10 <restart_handler+22>     mov     %rbp,%rbp
0x55555556f11 <restart_handler+25>     push    %r12
0x55555556f12 <restart_handler+28>     push    %r14
0x55555556f13 <restart_handler+31>     push    %r15
0x55555556f14 <restart_handler+34>     mov     %r10,%rbp
0x55555556f15 <restart_handler+37>     mov     %r10,%rbp
0x55555556f16 <restart_handler+40>     mov     %r10,%rbp
0x55555556f17 <restart_handler+43>     mov     %r10,%rbp
0x55555556f18 <restart_handler+46>     mov     %r10,%rbp
0x55555556f19 <restart_handler+49>     mov     %r10,%rbp
0x55555556f1a <restart_handler+52>     mov     %r10,%rbp
0x55555556f1b <restart_handler+55>     mov     %r10,%rbp
0x55555556f1c <restart_handler+58>     mov     %r10,%rbp
0x55555556f1d <restart_handler+61>     mov     %r10,%rbp
0x55555556f1e <restart_handler+64>     mov     %r10,%rbp
0x55555556f1f <restart_handler+67>     mov     %r10,%rbp
0x55555556f20 <restart_handler+70>     mov     %r10,%rbp
0x55555556f21 <restart_handler+73>     mov     %r10,%rbp
0x55555556f22 <restart_handler+76>     mov     %r10,%rbp
0x55555556f23 <restart_handler+79>     mov     %r10,%rbp
0x55555556f24 <restart_handler+82>     mov     %r10,%rbp
0x55555556f25 <restart_handler+85>     mov     %r10,%rbp
0x55555556f26 <restart_handler+88>     mov     %r10,%rbp
0x55555556f27 <restart_handler+91>     mov     %r10,%rbp
0x55555556f28 <restart_handler+94>     mov     %r10,%rbp
0x55555556f29 <restart_handler+97>     mov     %r10,%rbp
0x55555556f2a <restart_handler+100>    mov     %r10,%rbp
0x55555556f2b <restart_handler+103>    mov     %r10,%rbp
0x55555556f2c <restart_handler+106>    mov     %r10,%rbp
0x55555556f2d <restart_handler+109>    mov     %r10,%rbp
0x55555556f2e <restart_handler+112>    mov     %r10,%rbp
0x55555556f2f <restart_handler+115>    mov     %r10,%rbp
0x55555556f30 <restart_handler+118>    mov     %r10,%rbp
0x55555556f31 <restart_handler+121>    mov     %r10,%rbp
0x55555556f32 <restart_handler+124>    mov     %r10,%rbp
0x55555556f33 <restart_handler+127>    mov     %r10,%rbp
0x55555556f34 <restart_handler+130>    mov     %r10,%rbp
0x55555556f35 <restart_handler+133>    mov     %r10,%rbp
0x55555556f36 <restart_handler+136>    mov     %r10,%rbp
0x55555556f37 <restart_handler+139>    mov     %r10,%rbp
0x55555556f38 <restart_handler+142>    mov     %r10,%rbp

```

Рисунок 3 – Отображение ассемблерного кода

```

24 void restart_handler(int signal) {
25     restart_flag = 1; // установка флага перезапуска
26 }
27
28 int main(int argc, char *argv[])
29 {
30     signal(SIGUSR1, restart_handler);
31     //int seed = 1000000;
32     srand (time(NULL));
33     int row, col, cory = 0, cory = 0, diff, x = 0, y = 0, width, height, dor, dbf = 0, gc = 1, bn, wincounter = 0;
34     char move;
35     term_size(height, width);
36     int row, col, cory = 0, cory = 0, diff, x = 0, y = 0, width, height, dor, dbf = 0, gc = 1, bn, wincounter = 0;
37     welcome(width, height);
38     for (int i = 1; i < argc; i++)
39     {
40         if (strcmp(argv[i], "-dbf") == 0) {dbf = 1;}
41     }
42 }

```

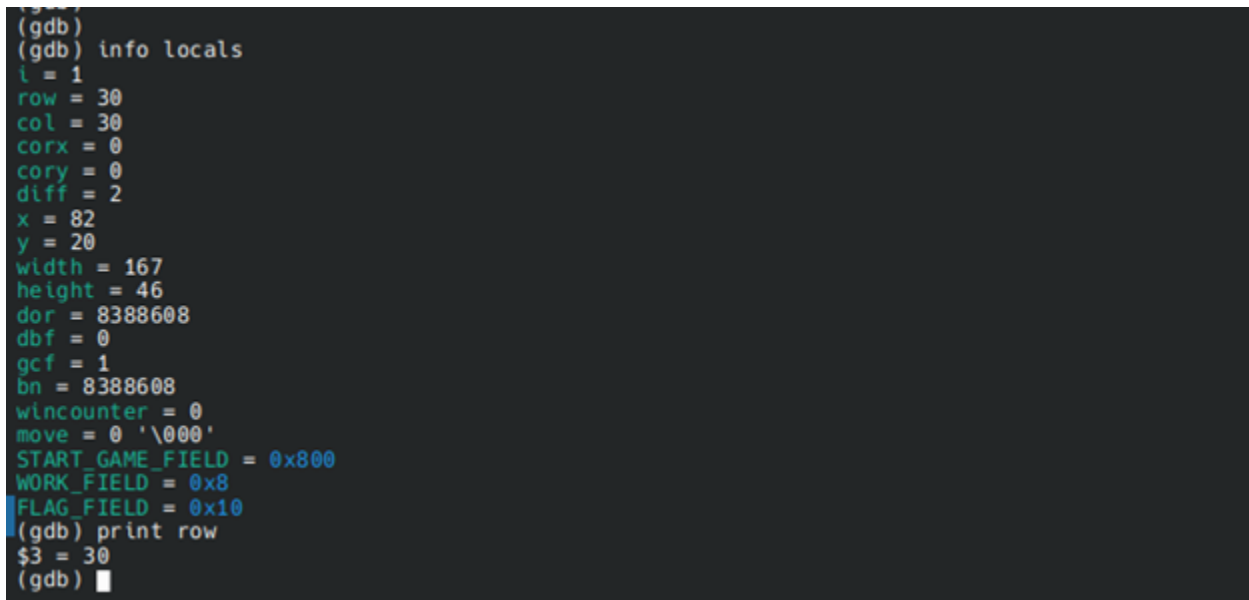
```

0x55555556f08 <restart_handler>      push    %rbp
0x55555556f09 <restart_handler+1>      mov     %rbp,%rbp
0x55555556f0a <restart_handler+4>      mov     %eax,%eax
0x55555556f0b <restart_handler+7>      movl    $0x1,%eax
0x55555556f0c <restart_handler+10>     pop     %rbp
0x55555556f0d <restart_handler+13>     ret
0x55555556f0e <restart_handler+16>     data16  data16 cs nops 0x0(0x0,0x0,1)
0x55555556f0f <restart_handler+19>     push    %rbp
0x55555556f10 <restart_handler+22>     mov     %rbp,%rbp
0x55555556f11 <restart_handler+25>     push    %r12
0x55555556f12 <restart_handler+28>     push    %r14
0x55555556f13 <restart_handler+31>     push    %r15
0x55555556f14 <restart_handler+34>     mov     %r10,%rbp
0x55555556f15 <restart_handler+37>     mov     %r10,%rbp
0x55555556f16 <restart_handler+40>     mov     %r10,%rbp
0x55555556f17 <restart_handler+43>     mov     %r10,%rbp
0x55555556f18 <restart_handler+46>     mov     %r10,%rbp
0x55555556f19 <restart_handler+49>     mov     %r10,%rbp
0x55555556f1a <restart_handler+52>     mov     %r10,%rbp
0x55555556f1b <restart_handler+55>     mov     %r10,%rbp
0x55555556f1c <restart_handler+58>     mov     %r10,%rbp
0x55555556f1d <restart_handler+61>     mov     %r10,%rbp
0x55555556f1e <restart_handler+64>     mov     %r10,%rbp
0x55555556f1f <restart_handler+67>     mov     %r10,%rbp
0x55555556f20 <restart_handler+70>     mov     %r10,%rbp
0x55555556f21 <restart_handler+73>     mov     %r10,%rbp
0x55555556f22 <restart_handler+76>     mov     %r10,%rbp
0x55555556f23 <restart_handler+79>     mov     %r10,%rbp
0x55555556f24 <restart_handler+82>     mov     %r10,%rbp
0x55555556f25 <restart_handler+85>     mov     %r10,%rbp
0x55555556f26 <restart_handler+88>     mov     %r10,%rbp
0x55555556f27 <restart_handler+91>     mov     %r10,%rbp
0x55555556f28 <restart_handler+94>     mov     %r10,%rbp
0x55555556f29 <restart_handler+97>     mov     %r10,%rbp
0x55555556f2a <restart_handler+100>    mov     %r10,%rbp
0x55555556f2b <restart_handler+103>    mov     %r10,%rbp
0x55555556f2c <restart_handler+106>    mov     %r10,%rbp
0x55555556f2d <restart_handler+109>    mov     %r10,%rbp
0x55555556f2e <restart_handler+112>    mov     %r10,%rbp
0x55555556f2f <restart_handler+115>    mov     %r10,%rbp
0x55555556f30 <restart_handler+118>    mov     %r10,%rbp
0x55555556f31 <restart_handler+121>    mov     %r10,%rbp
0x55555556f32 <restart_handler+124>    mov     %r10,%rbp
0x55555556f33 <restart_handler+127>    mov     %r10,%rbp
0x55555556f34 <restart_handler+130>    mov     %r10,%rbp
0x55555556f35 <restart_handler+133>    mov     %r10,%rbp
0x55555556f36 <restart_handler+136>    mov     %r10,%rbp
0x55555556f37 <restart_handler+139>    mov     %r10,%rbp
0x55555556f38 <restart_handler+142>    mov     %r10,%rbp

```

Рисунок 4 – Отображение исходного и ассемблерного кода одновременно

2. **Команда info.** Команда info позволяет получать различную информацию о программе, включая информацию о регистрах, переменных, точках останова и других. Можно использовать команды, такие как info registers (показать значения регистров), info locals (показать локальные переменные) и info breakpoints (показать список точек останова), для получения нужной информации.
3. **Команда print.** Команда print позволяет получать значение переменных.

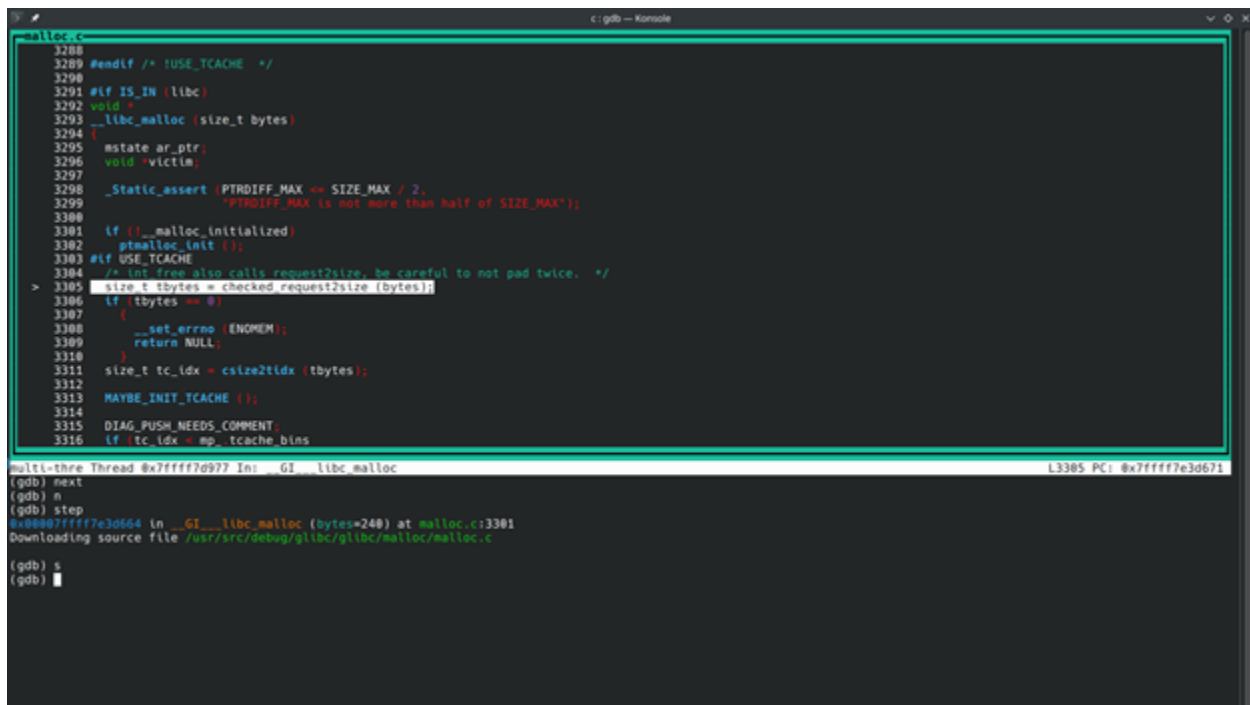


```
(gdb)
(gdb) info locals
i = 1
row = 30
col = 30
corx = 0
cory = 0
diff = 2
x = 82
y = 20
width = 167
height = 46
dor = 8388608
dbf = 0
gcf = 1
bn = 8388608
wincounter = 0
move = 0 '\000'
START_GAME_FIELD = 0x800
WORK_FIELD = 0x8
FLAG_FIELD = 0x10
(gdb) print row
$3 = 30
(gdb)
```

Рисунок 5 – Пример использования команды print и info

3.3. Переход по коду в отладчике с помощью пошаговых команд

В среде отладки **GDB** есть возможность перемещаться по коду с помощью команд **next**, **step**.



```
3288 #endif /* !USE_TCACHE */
3289 #endif
3290 #if IS_IN (libc)
3291 void *
3292 __libc_malloc (size_t bytes)
3293 {
3294     mstate ar_ptr;
3295     void *victim;
3296     _Static_assert (PTRDIFF_MAX <= SIZE_MAX / 2,
3297                     "PTRDIFF_MAX is not more than half of SIZE_MAX");
3298     if (!__malloc_initialized)
3299         ptmalloc_init ();
3300 #if USE_TCACHE
3301     /* int free also calls request2size, be careful to not pad twice. */
3302     size_t tbytes = checked_request2size (bytes);
3303     if (tbytes == 0)
3304     {
3305         __set_errno (ENOMEM);
3306         return NULL;
3307     }
3308     size_t tc_idx = csize2tidx (tbytes);
3309     MAYBE_INIT_TCACHE ();
3310     DIAG_PUSH_NEEDS_COMMENT
3311     if (tc_idx < mp_tcache_bins
```

Multi-Thread Thread 0x7ffff7d977 In: __GI___libc_malloc L3305 PC: 0x7ffff7e3d671

(gdb) next
(gdb) n
(gdb) step
0x00007ffff7e3d664 in __GI___libc_malloc (bytes=240) at malloc.c:3301
Downloading source file /usr/src/debug/glibc/glibc/malloc/malloc.c
(gdb) s
(gdb) █

Рисунок 5 – Пример использования команд перехода по коду

4. Описание программы

Разработанная программа состоит из нескольких модулей, которые реализовались в следующем порядке:

1. `saper.c` — основной файл программы, в котором запускается тело игры.
2. `term_tools.c` — файл с упрощёнными интерфейсами для работы программой с терминалом.
3. `gamefn.c` — файл со всеми игровыми функциями
4. `decorations.c` — файл с декорациями.

Программа **saper** является основным модулем программы. При запуске происходит вывод заставки-приветствия и запуск **фоновой** музыки используя стороннюю программу **mpv** с аргументами **--loop=inf --quiet --no-video --no-**

terminal 1.mp3, после чего с пользователем запускается диалог о настройке игры. Вход в саму игру осуществляется после завершения диалога. Если пользователь хочет пропустить диалог, он может запустить программу с аргументом **-d** (default). Этот аргумент задаст стандартные параметры (Размер поля: 30x30, сложность: нормальная). В игре предусмотрена система перемещения курсора клавишами перемещения. Пользователь может переместить курсор на желаемую позицию и открыть клетку. Генерация бомб происходит в момент открытия первой клетки. Это сделано для того, чтобы пользователь не проиграл при первом открытии клетки.\

Далее на рисунке 6 предоставлена схема взаимодействия программы.

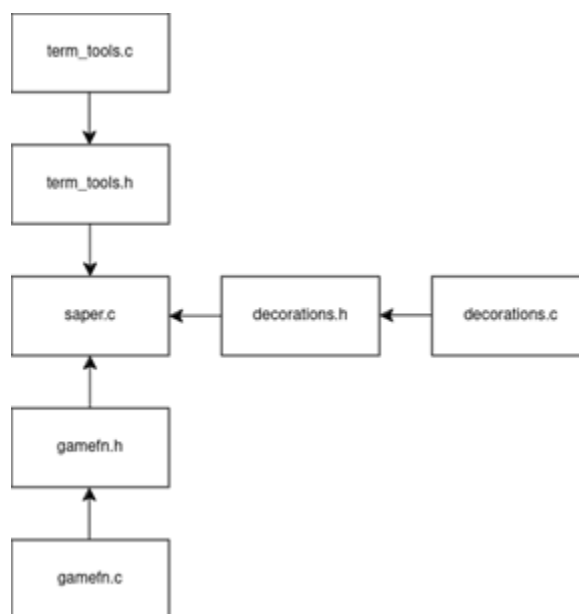


Рисунок 6 – схема взаимодействия программы.

5. Схема программы

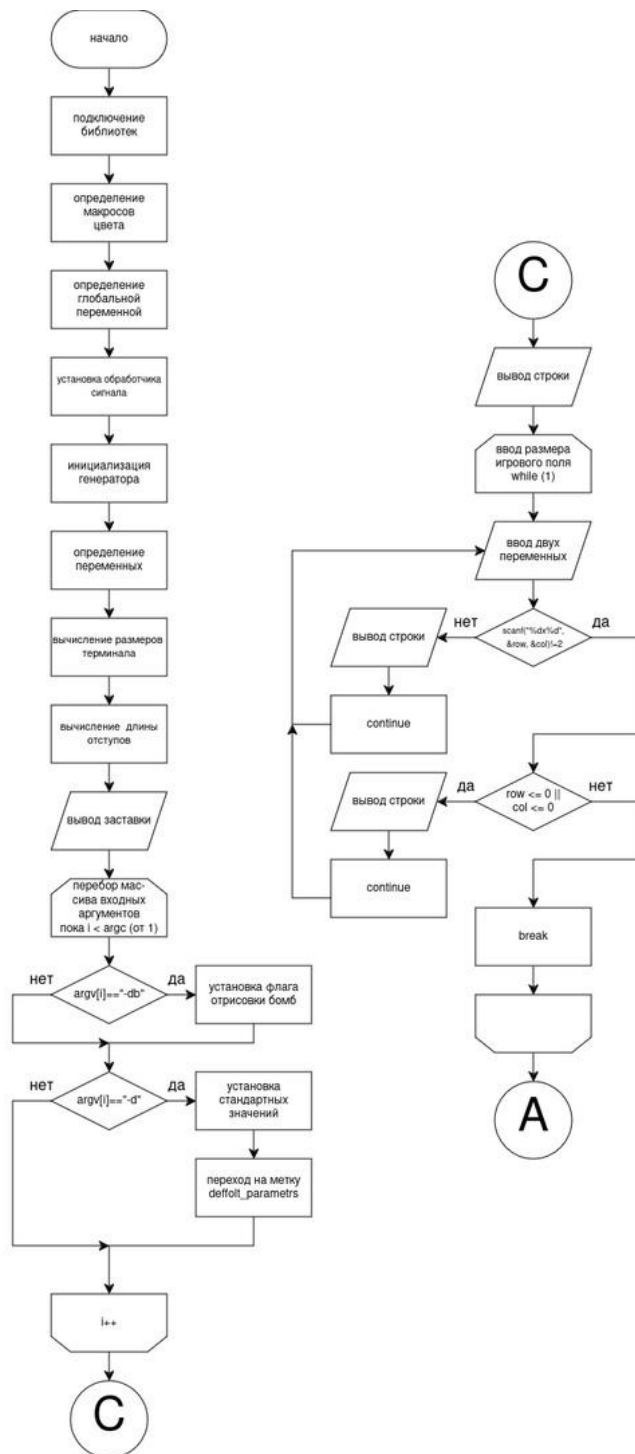


Рисунок 7 – Схема программы

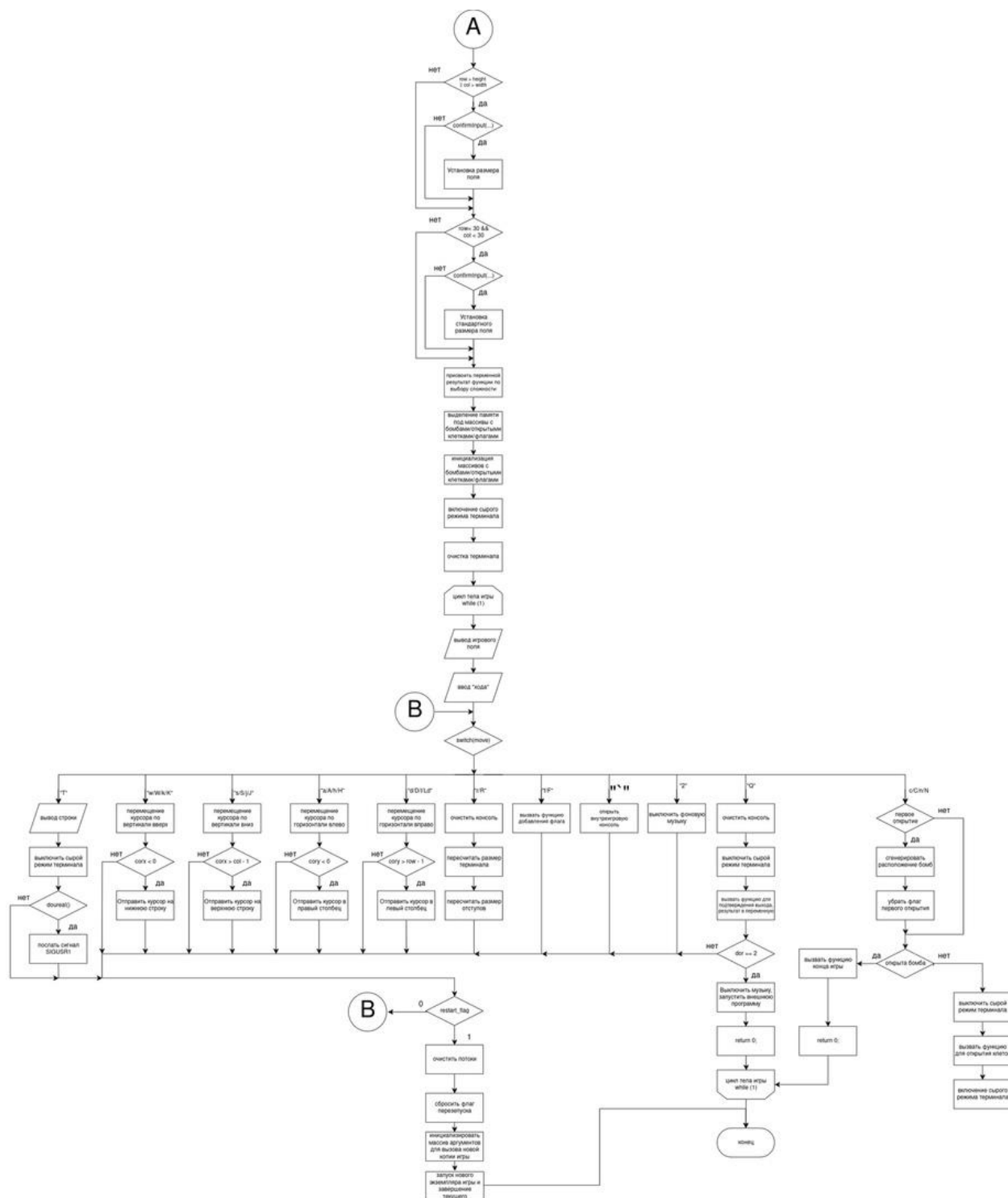
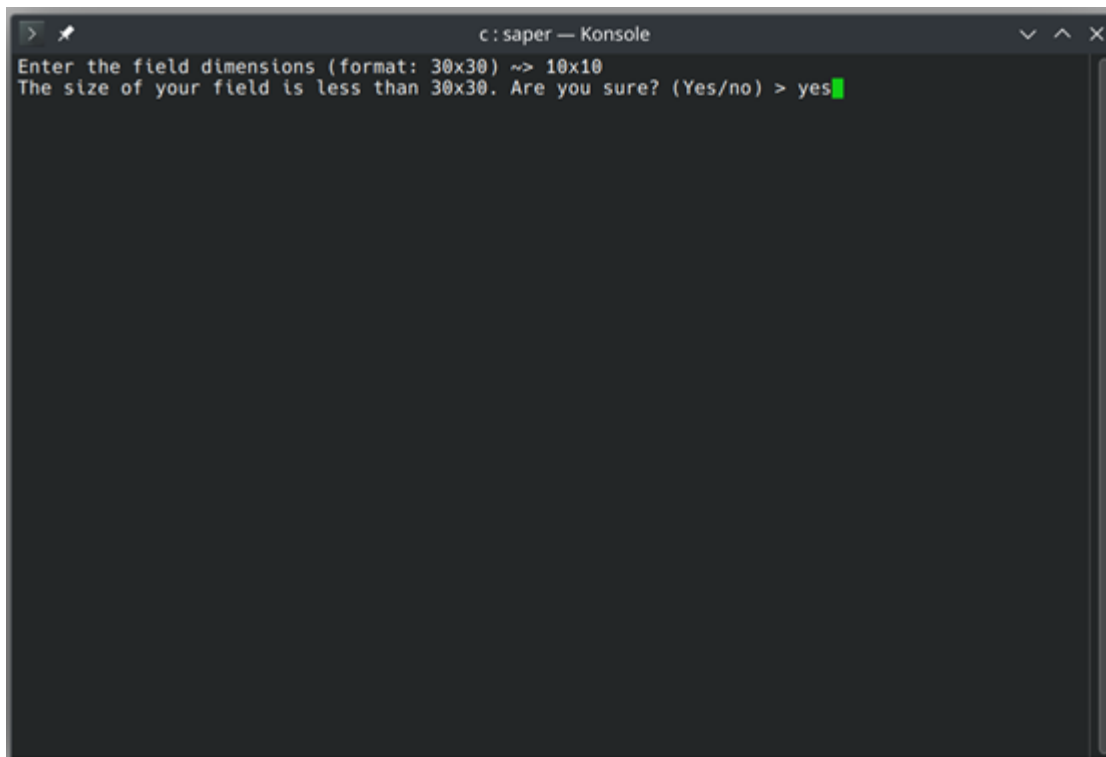


Рисунок 8 – Схема программы

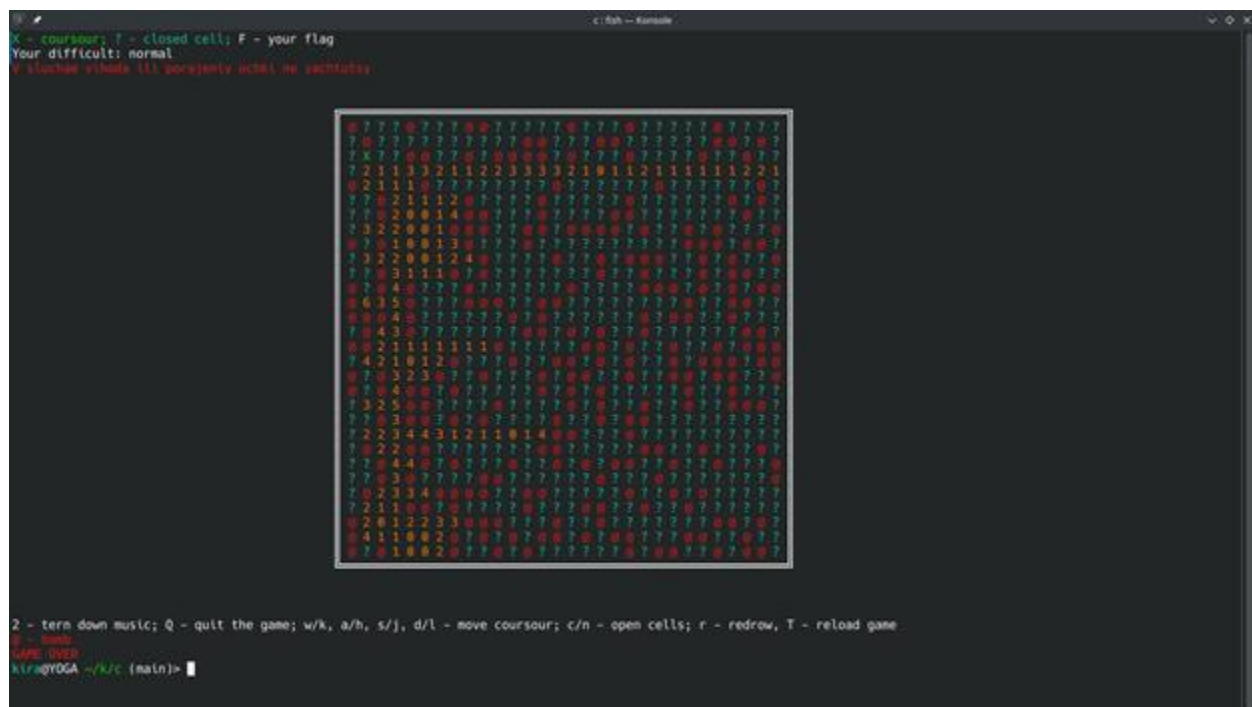
6. Руководство пользователя

6.1. Вступительный диалог

Программа saper – игра с консольным интерфейсом (**CLI**). Программа имеет интуитивно понятный интерфейс. После заставки-приветствия запускается диалог с пользователем (с возможностью пропустить используя аргумент **-d** при запуске).



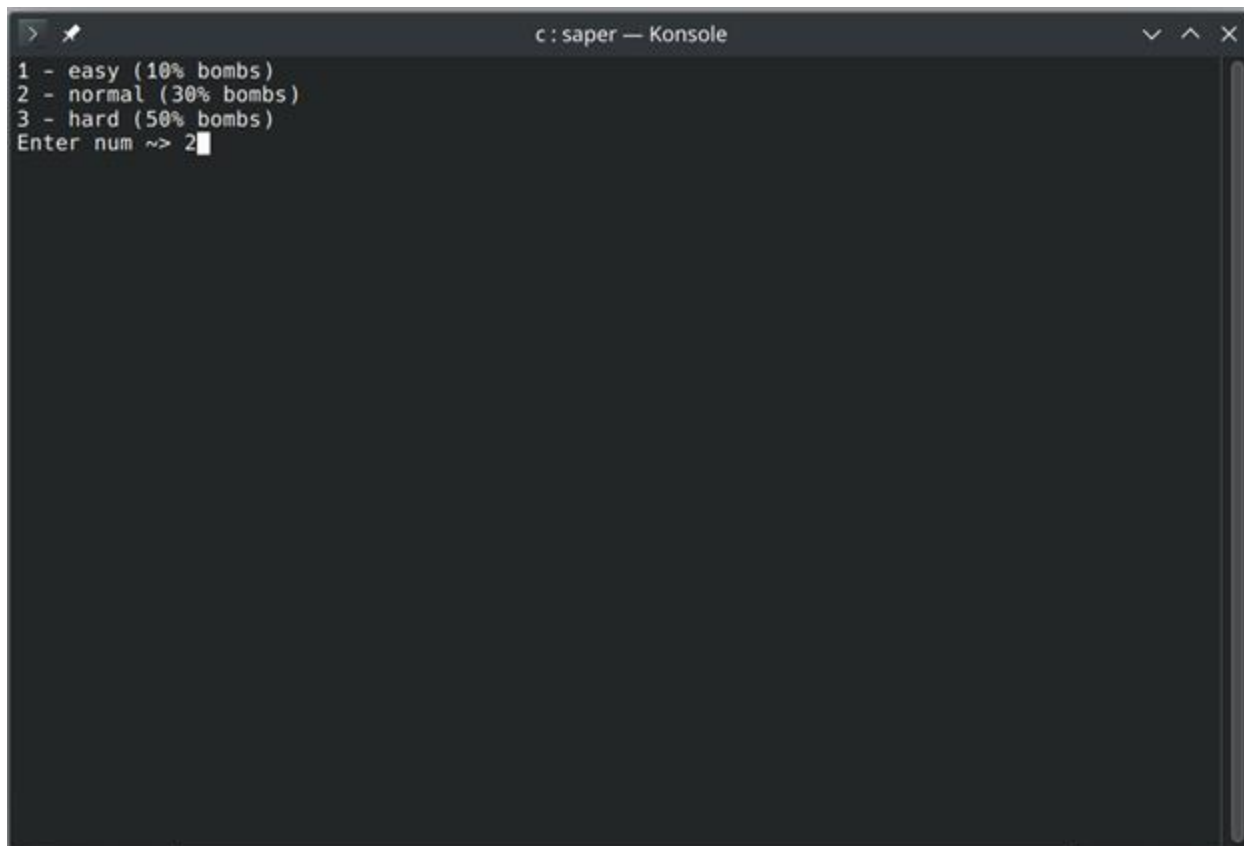
```
c : saper — Konsole
Enter the field dimensions (format: 30x30) ~> 10x10
The size of your field is less than 30x30. Are you sure? (Yes/no) > yes
```



На этом этапе пользователю будет предложено ввести размер поля в формате **XxY**. Так же при необходимости будут заданы уточняющие вопросы:

- Уверен ли пользователь, что размер поля меньше рекомендуемого (30x30).
- Уверен ли пользователь, что хочет запустить игру с размером поля больше окна терминала.

После пользователю будет предложено указать желаемую сложность.

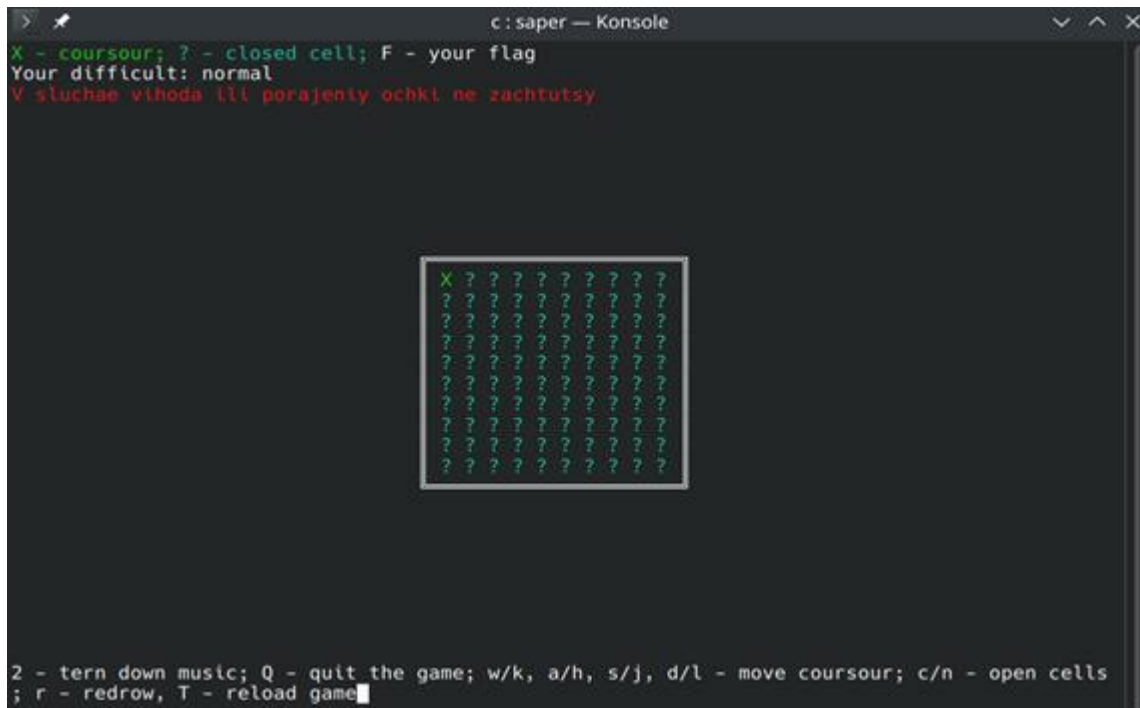


```
c : saper — Konsole
1 - easy (10% bombs)
2 - normal (30% bombs)
3 - hard (50% bombs)
Enter num ~> 2
```

Пользователь может выбрать желаемую сложность введя цифру этой сложности. После чего пользователь будет допущен в игровой процесс.

6.2. Игровой процесс

По мимо поля также будет и отображена служебная информации, такая как: легенда поля; команды, доступные пользователю.



The screenshot shows a terminal window titled "c: saper — Konsole". The game interface includes a legend at the top: "X - coursour; ? - closed cell; F - your flag", the difficulty level "Your difficult: normal", and a warning in Russian "V sluchae vihoda ili porajenty ochki ne zachtutsy". The main game area is a 10x10 grid of characters. The first cell (0,0) contains 'X', and the rest are '?'. A white rectangular highlight is around the first two rows. At the bottom, a list of controls is shown: "2 - tern down music; Q - quit the game; w/k, a/h, s/j, d/l - move coursour; c/n - open cells; r - redraw, T - reload game".

```
> c: saper — Konsole
X - coursour; ? - closed cell; F - your flag
Your difficult: normal
V sluchae vihoda ili porajenty ochki ne zachtutsy

  X ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ?

2 - tern down music; Q - quit the game; w/k, a/h, s/j, d/l - move coursour; c/n - open cells
; r - redraw, T - reload game
```

Пользователь может передвигать курсор как клавишами **wasd**, так и раскладкой **vim**. Открытие клеток происходит по нажатию клавиши **c** или **n**. При первом открытии будет сгенерировано расположение бомб и их количество относительно выбранной ранее сложности.

После успешного завершения игры пользователя ожидает экран завершения и результат его игры

```
>  c: fish — Konsole
You've Won!!!!
Your score: 23

< POBEDAAA, horosh!! >
-----
      \      ^__^
       \      (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

kira@YOGA ~/k/c (main)>
```

В случае поражения программа покажет пользователю расположение всех бомб и завершит игру.

The screenshot shows a terminal window with a Minesweeper game. The game board is a 20x20 grid. The top part of the grid shows numbers and mines (X). The bottom part of the grid shows the command prompt and a list of commands.

```

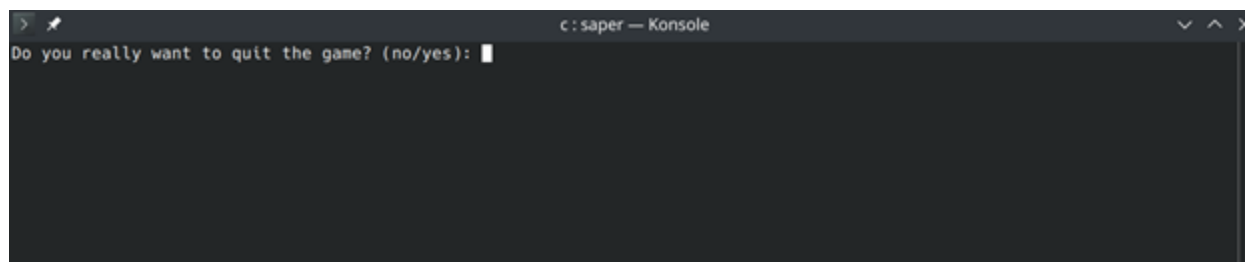
c:\Bsh - Konsole
x = coursour; f = closed cell; F = your flag
Your difficult: normal
x kluchee vkhude lli porajeniy wchali ne zachlutsy

  7 7 7 0 7 7 7 0 7 7 7 7 0 7 7 7 0 7 7 7 7
  7 0 7 7 7 7 7 7 7 7 0 7 7 7 0 7 7 2 2 7 0 7
  7 X 7 7 0 7 7 0 7 0 0 0 7 0 7 7 7 0 7 7 7 7
  7 2 1 1 3 3 2 1 1 2 2 3 3 3 2 1 0 1 1 2 1 1
  0 2 1 1 0 7 7 7 7 7 7 7 0 7 7 7 7 7 7 7 7
  7 7 0 2 1 1 1 2 0 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 0 2 0 0 1 4 0 7 7 7 7 7 7 0 7 7 7 7 7
  7 3 2 2 0 0 1 0 0 7 7 7 7 7 7 7 7 7 7 7 7
  7 3 1 0 0 1 3 0 0 7 7 7 7 7 7 7 7 7 7 7 7
  7 3 2 0 0 1 2 4 0 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 3 1 1 4 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 4 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  0 6 1 5 7 7 7 0 0 7 7 7 7 7 7 7 7 7 7 7
  0 0 4 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 0 4 3 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  0 0 2 1 1 1 1 1 1 1 0 7 7 7 7 7 7 7 7 7
  7 4 2 1 0 1 2 0 7 7 7 7 7 7 7 7 7 7 7 7
  0 7 0 3 2 3 0 7 7 7 7 7 7 7 7 7 7 7 7 7
  0 7 0 4 0 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 3 2 5 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 7 3 0 0 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 2 2 4 4 3 1 7 7 7 7 7 7 7 7 7 7 7
  7 7 0 2 2 0 0 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 7 4 4 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 7 3 0 7 7 7 7 7 7 7 7 7 7 7 7 7 7
  7 7 2 3 3 4 0 0 0 7 7 7 7 7 7 7 7 7 7
  7 2 1 1 0 0 7 7 7 7 7 7 7 7 7 7 7 7 7
  0 2 0 1 2 2 3 3 0 0 0 7 7 7 7 7 7 7 7
  0 4 1 1 0 0 2 0 7 7 7 7 7 7 7 7 7 7 7
  7 0 1 0 0 2 0 7 7 7 7 7 7 7 7 7 7 7 7

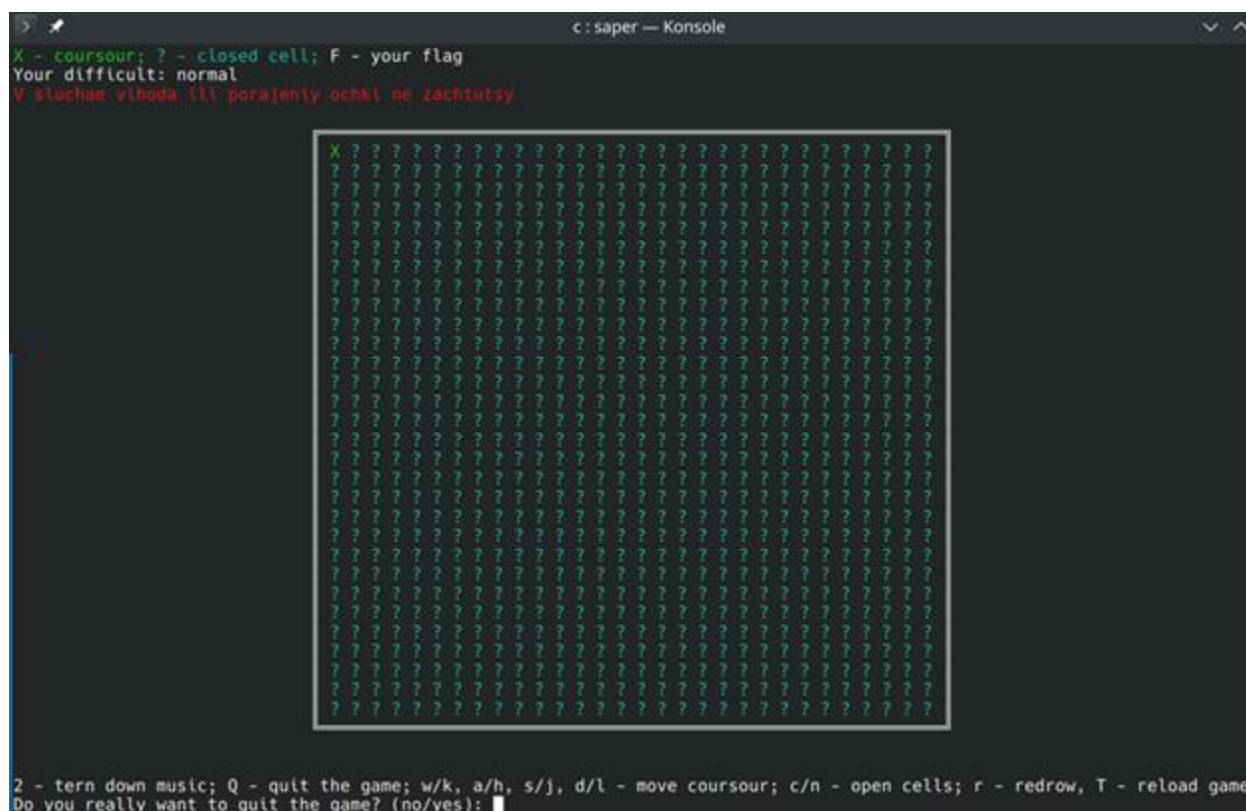
2 - turn down music; Q - quit the game; w/x, a/h, s/j, d/l - move coursour; c/n - open cells; r - redraw, T - reload game
c - Read
GAME OVER
kirs@YOGA ~/K/c (main)>

```

Пользователь может досрочно завершить игру нажав Q, что запустит диалог о завершении игры.



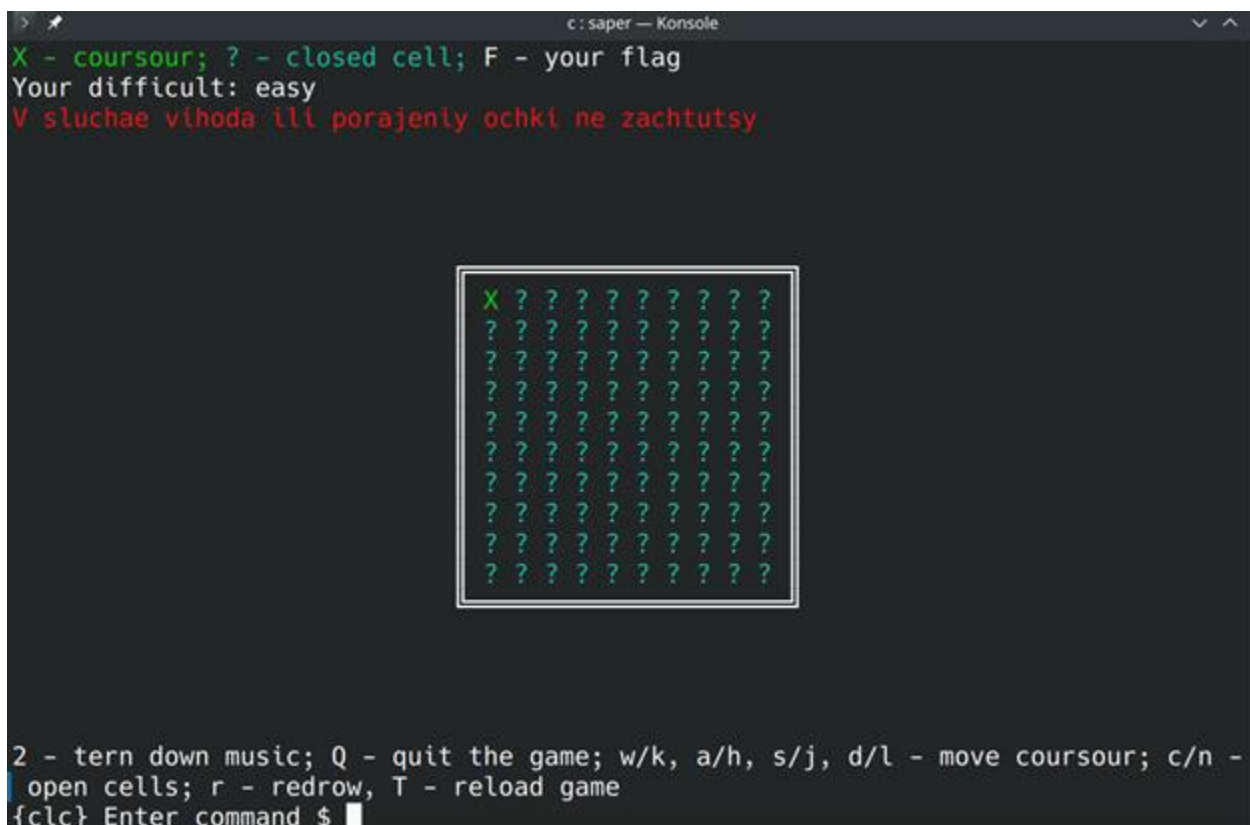
Пользователь также может перезапустить игру нажав T, что запустит диалог о выходе из игры.



В случае досрочного завершения, поражения или перезапуска игры счет пользователя не будет сохранен.

Также пользователь по желанию может выключить фоновую музыку клавишей 2. Это действие перманентно выключит музыку без возможности запустить ее снова в текущем экземпляре игры.

Также в игре предусмотрена реализация игровой **консоль**. **Консоль** открывается клавишей ` , это действие откроет приглашение к вводу команды в игровую консоль.



```
c : saper — Konsole
X - coursour; ? - closed cell; F - your flag
Your difficult: easy
V sluchae vihoda ili porajenly ochki ne zachtutsy

  X ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?
  ? ? ? ? ? ? ? ? ?

2 - tern down music; Q - quit the game; w/k, a/h, s/j, d/l - move coursour; c/n -
open cells; r - redraw, T - reload game
{clc} Enter command $
```

Пользователь может ввести 3 команды:

1. `drowb` — команда для установки флага отрисовки бомб (флаг также устанавливается при запуске игры с аргументом **-db**)
2. `ndrowb` — Команда для снятия флага отрисовки бомб
3. `lead` — Команда для отображения лучших результатов

Заключение

При выполнении данной курсовой работы были получены навыки разработки многомодульных программ. Была освоена работа с файлами, изучены функции работы с консолью и оперативной памятью. Также были получены основные навыки отладки и тестирования программ в среде **GDB** и программирования в среде **Neovim** на языках Си и Assembler.

В рамках курсовой работы была разработана игра “Сапер”. Она имеет набор всех необходимых механик для приятной игры.

В дальнейшем программу можно улучшить, путём добавления новых цветовых схем, добавление управления с помощью мыши, усложнить систему генерации расположения бомб, что подарит пользователю более приятный игровой опыт.

Приложение А

Листинг программы

Файл *saper.c*

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <termios.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include "libs/term_tools.h"
#include "libs/decorations.c"
#include "libs/gamefn.h"

// обозначение светов
#define Reset "\x1b[0m"//белый
#define ColRed "\x1b[31m"
#define ColCors "\x1b[32m"
#define ColNum "\x1b[33m"
#define ColAnn "\x1b[36m"
//#define reset_curs printf("")

int restart_flag = 0;

void restart_handler(int signal) {
    restart_flag = 1; // установка флага перезапуска
}

int main(int argc, char *argv[])
{
    signal(SIGUSR1, restart_handler);
    //int seed = time(NULL);
    srand(time(NULL));
```

```

int row, col, corx = 0, cory = 0, diff, x = 0, y = 0, width, height, dor, dbf = 0, gcf
= 1, bn, wincounter = 0;
char move;
term_size(&height, &width);
indent(&width, &height, &x, &y, col, row);
welcome(&width, &height);

for (int i = 1; i < argc; i++)
{
    if (strcmp(argv[i], "-db") == 0) {dbf = 1;}
    if (strcmp(argv[i], "-d") == 0)
    {
        row = 30;
        col = 30;
        diff = 2;
        //dbf = 1;

        goto deffolt_parametrs;
    }
}

printf("Enter the field dimensions (format: 30x30) ~> ");
while (1)
{
    if (scanf("%dx%d", &row, &col) != 2)
    {
        printf("Input ERROR, please enter again (format: 30x30) ~> ");
        //while (getchar() != '\n'); // очищаем буфер ввода
        continue;
    }
    if (row <= 0 || col <= 0)
    {
        printf("Input ERROR, please enter again (format: 30x30) ~> ");
        continue;
    }
    break;
}

```

```

if (row > height || col > width)
{
    if (confirmInput(width, height, "The size of your field is larger than the size of
the terminal. Are you sure? (Yes/no) ~> ", width / 4, height / 2) == 0)
    {
        row = height / 4;
        col = width / 4;
    }
} else if (row < 30 && col < 30)
{
    if (confirmInput(width, height, "The size of your field is less than 30x30. Are
you sure? (Yes/no) > ", 30, 30) == 0)
    {
        row = 30;
        col = 30;
    }
}

diff = choseDifficulty();
//allocation memmory to fields
deffolt_parametrs: diff = diff;
char **START_GAME_FIELD = (char **)malloc(col * sizeof(char *));
for (int i = 0; i < col; i++)
{
    START_GAME_FIELD[i] = (char *)malloc(row * sizeof(char));
}

for (int i = 0; i < col; i++)
{
    for (int j = 0; j < row; j++)
    {
        START_GAME_FIELD[i][j] = '9';
    }
}

char **WORK_FIELD = (char **)malloc(col * sizeof(char *));

```

```

for (int i = 0; i < col; i++)
{
    WORK_FIELD[i] = (char *)malloc(row * sizeof(char));
}
char **FLAG_FIELD = (char **)malloc(row * sizeof(char *));
for (int i = 0; i < row; i++)
{
    FLAG_FIELD[i] = (char *)malloc(col * sizeof(char));
}

for (int i = 0; i < col; i++)
{
    for (int j = 0; j < row; j++)
    {
        START_GAME_FIELD[i][j] = 9;
        WORK_FIELD[i][j] = 9;
        FLAG_FIELD[j][i] = 9;
    }
}

enableRawMode();
system("clear");

while(1)
{
    dryFd(START_GAME_FIELD, WORK_FIELD, FLAG_FIELD, &y, &x, row,
col, corx, cory, diff, dbf, &height);

    scanf("%c", &move);
    switch(move)
    {
        case 'w':
        case 'W':
        case 'k':
        case 'K':
            corx--;
            if (corx < 0){corx = col-1;}

```

```

        break;
    case 's':
    case 'S':
    case 'j':
    case 'J':
        corx++;
        if (corx > col - 1){corx= 0;}
        break;
    case 'a':
    case 'A':
    case 'h':
    case 'H':
        cory--;
        if (cory < 0){cory = row-1;}
        break;
    case 'd':
    case 'D':
    case 'l':
    case 'L':
        cory++;
        if (cory > row - 1){cory= 0;}
        break;
    case 'r':
    case 'R':
        system("clear");
        term_size(&height, &width);
        indent(&width, &height, &x, &y, col, row);
        break;
    case 'Q':
        system("clear");
        disableRawMode();
        dor = doureal();
        if (dor == 2)
        {
            //printf("%d\n", dor);
            system("killall mpv; clear; cowsay spasiba za igru!!");
            //printf("\n");

```

```

        return 0;
    }
    else if (dor == 1)
    {
        enableRawMode();
        break;
    }
    break;
case 'f':
case 'F':
    addFlag(FLAG_FIELD, corx, cory);
    break;
case '`':
    clc__console__(&dbf);
    break;
case '2':
    system("killall mpv;clear");
    break;
case 'c':
case 'C':
case 'n':
case 'N':
    if (gcf){bn = genCode(START_GAME_FIELD, diff, col, row, corx, cory); gcf
=0;}
    if (START_GAME_FIELD[corx][cory] == '*')
    {
        EndGame(START_GAME_FIELD, WORK_FIELD, FLAG_FIELD, &y, &x,
row, col, corx, cory, diff, &height, ColRed"\nGAME OVER\n" Reset);
        return 0;
    }
    disableRawMode();
    open_cell(START_GAME_FIELD, WORK_FIELD, cory, corx, col, row, bn,
&wincounter, diff);
    enableRawMode();
    break;
case 'T':
    printf("\n");

```

```

    disableRawMode();
    if (doureal())
    {
        raise(SIGUSR1);
    }
}
if (restart_flag) {
    fflush(stdin);
    fflush(stdout);
    restart_flag = 0; // сброс флага перезапуска
    char *argv[] = {"/saper", NULL}; // массив аргументов для вызова новой
копии игры
    execv(argv[0], argv); // запуск новой копии игры и завершение текущего
процесса
}
}
//system("killall mpv");
}

```

Файл *asm-x86_64.c*

```

#include <stdlib.h>

int main() {
    char* cmd = "mpv --loop=inf --quiet --no-video --no-terminal 1.mp3";
    int result;

    asm volatile (
        "mov %1, %%rdi\n\t" // Переместить адрес cmd в rdi (1-й аргумент для
system)
        "call *%2\n\t"      // Вызвать функцию system
        "mov %%eax, %0"     // Сохранить результат (возвращаемое значение
system) в result
        : "=r" (result)    // Операнд вывода
        : "r" (cmd), "r" (system) // Операнды ввода
    );
}

```

```

        : "rdi", "rax"    // Модифицируемые регистры
    );

    if (result != 0) {
        return 1;
    }

    return 0;
}

```

Файл `gamefn.c`

```

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <termios.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <stdio.h>
#include <string.h>

```

```

// обозначение светов
#define Reset "\x1b[0m"//белый
#define ColRed "\x1b[31m"
#define ColCors "\x1b[32m"
#define ColNum "\x1b[33m"
#define ColQe "\x1b[36m"
//#define reset_curs printf("")

#define CUSTOM 0
#define EASY 1
#define NORMAL 2
#define HARD 3
#define VERY_HARD 4
#define IMPOSSIBLE 5

void dryFd(char **START_GAME_FIELD, char **WORK_FIELD, char
**FLAG_FIELD, int *y, int *x, int row, int col, int corx, int cory, int diff, int dbf,
int *width);

void add_score(int new_score)
{
    FILE *file = fopen("scores.txt", "r+");

    //if (file == NULL) {
        //return;
    //}

    int minNumber = new_score;
    int minPosition = 0;

    int currentNumber;
    int position = 0;

    rewind(file);
    while (fscanf(file, "%d", &currentNumber) == 1)
    {
        if (currentNumber < minNumber)

```

```

        {
            minNumber = currentNumber;
            minPosition = position;
        }
        position++;
    }

    fseek(file, minPosition * sizeof(int), SEEK_SET);

    fprintf(file, "%d ", new_score);

    fclose(file);
}

void EndGame(char **START_GAME_FIELD, char **WORK_FIELD, char
**FLAG_FIELD, int *y, int *x, int row, int col, int corx, int cory, int diff, int
*height, char* msg)
{
    system("clear");
    dryFd(START_GAME_FIELD, WORK_FIELD, FLAG_FIELD, y, x, row, col,
corx, cory, diff, 1, height);
    printf("%s", msg);
    exit(0);
}

void open_cell(char **START_GAME_FIELD, char **WORK_FIELD, int cory,
int corx, int col, int row, int bn, int *wincounter, int diff)
{
    int aw = 0;
    int dx[] = {-1, 0, 1, 0, -1, -1, 1, 1};
    int dy[] = {0, 1, 0, -1, -1, 1, -1, 1};

    if (cory < 0 || corx > col || corx < 0 || cory > row ||
START_GAME_FIELD[corx][cory] == '*')
    {

```

```

    return;
}

//вправо вниз
for (int tcrx = corx; tcrx < col; tcrx++)
{
    for (int tcry = cory; tcry < row; tcry++)
    {
        int count = 0;
        if (START_GAME_FIELD[tcrx][tcry]=='*')
        {
            break;
        }
        for (int i = 0; i < 8; i++)
        {
            int tx = tcrx + dx[i];
            int ty = tcry + dy[i];

            if (tx >= 0 && tx < col && ty >= 0 && ty < row &&
START_GAME_FIELD[tx][ty] == '*')
            {
                count++;
            }
        }

        if (WORK_FIELD[tcrx][tcry] == 9)
        {
            (*wincounter)++;
            WORK_FIELD[tcrx][tcry] = count;
        }
        if (tcrx+1 < col && START_GAME_FIELD[tcrx+1][cory] == '*'){aw = 1;
break;}
    }
    if (aw){aw = 0; break;}
}

```

```

//ВЛЕВО ВНИЗ
for (int tcrx = corx; tcrx < col; tcrx++)
{
    for (int tcry = cory; tcry > 0; tcry--)
    {
        int count = 0;
        if (START_GAME_FIELD[tcrx][tcry]=='*')
        {
            break;
        }
        for (int i = 0; i < 8; i++)
        {
            int tx = tcrx + dx[i];
            int ty = tcry + dy[i];

            if (tx >= 0 && tx < col && ty >= 0 && ty < row &&
START_GAME_FIELD[tx][ty] == '*')
            {
                count++;
            }
        }
        if (WORK_FIELD[tcrx][tcry] == 9)
        {
            (*wincounter)++;
            WORK_FIELD[tcrx][tcry] = count;
        }
        if (tcrx+1 < col && START_GAME_FIELD[tcrx+1][cory] == '*'){aw = 1;
break;}
    }
    if (aw){aw = 0; break;}
}

```

```

//ВПРАВО ВВЕРХ
for (int tcrx = corx; tcrx > 0; tcrx--)
{
    for (int tcry = cory; tcry < row; tcry++)
    {

```

```

int count = 0;
if (START_GAME_FIELD[tcrx][tcry]=='*')
{
    break;
}
for (int i = 0; i < 8; i++)
{
    int tx = tcrx + dx[i];
    int ty = tcry + dy[i];

    if (tx >= 0 && tx < col && ty >= 0 && ty < row &&
START_GAME_FIELD[tx][ty] == '*')
    {
        count++;
    }
}
if (WORK_FIELD[tcrx][tcry] == 9)
{
    (*wincounter)++;
    WORK_FIELD[tcrx][tcry] = count;
}
if (tcrx+1 < col && START_GAME_FIELD[tcrx-1][tcry] == '*'){aw = 1;
break;}//////////
}
if (aw){aw = 0; break;}
}

//влево вверх
for (int tcrx = corx; tcrx > 0; tcrx--)
{
    for (int tcry = cory; tcry > 0; tcry--)
    {
        int count = 0;
        if (START_GAME_FIELD[tcrx][tcry]=='*')
        {
            break;
        }
    }
}

```

```

for (int i = 0; i < 8; i++)
{
    int tx = tcrx + dx[i];
    int ty = tcry + dy[i];

    if (tx >= 0 && tx < col && ty >= 0 && ty < row &&
START_GAME_FIELD[tx][ty] == '*')
    {
        count++;
    }
}
if (WORK_FIELD[tcrx][tcry] == 9)
{
    (*wincounter)++;
    WORK_FIELD[tcrx][tcry] = count;
}
if (tcrx+1 < col && START_GAME_FIELD[tcrx-1][tcry] == '*'){aw = 1;
break;}////////////////////
}
if (aw){aw = 0; break;}
}

```

Рисунок 2 – отображение исходного кода

```

if (bn <= *wincounter)
{
    int winc = *wincounter;
    //if (diff == 1){winc}
    if (diff == 2){winc = winc * 5;}
    else if (diff == 3){winc = winc * 9;}
    add_score(winc);
    system("clear");
    printf("You've Won!!!!\nYour score: %d\n", winc);
    system("cowsay POBEDAAA, horosh!!");
    //EndGame(START_GAME_FIELD, WORK_FIELD, FLAG_FIELD, int *y,
int *x, int row, int col, int corx, int cory, int diff, int *height, char *msg);
    exit(0);
}
}

```

```

int choseDifficulty()
{
    int diff = 9;
    while (diff != CUSTOM && diff != EASY && diff != NORMAL && diff !=
HARD && diff != VERY_HARD && diff != IMPOSSIBLE)
    {
        system("clear");
        printf("1 - easy (10%% bombs)\n2 - normal (30%% bombs)\n3 - hard (50%%
bombs)\nEnter num ~> ");
        fflush(stdin);
        scanf("%d", &diff);
    }
    return diff;
}

```

```

void addFlag(char **FLAG_FIELD, int cory, int corx)
{
    if (FLAG_FIELD[corx][cory] != '1')
    {
        FLAG_FIELD[corx][cory] = '1';
    }
    else FLAG_FIELD[corx][cory] = '0';
}

```

```

int genCode(char **START_GAME_FIELD, int diff, int row, int col, int corx, int
cory)
{
    int totalCells = row * col;
    int numBombs = 0;
    double threshold = 0.0;

    switch (diff) {

```

```

case EASY:
    threshold = 0.1;
    break;
case NORMAL:
    threshold = 0.3;
    break;
case HARD:
    threshold = 0.5;
    break;
}

numBombs = totalCells * threshold;

for (int i = 0; i < numBombs; )
{
    int x = (rand() % row);
    int y = (rand() % col);
    if (x < 0 || x >= row || y < 0 || y >= col)
    {
        continue;
    }
    else
    {
        if (START_GAME_FIELD[x][y] != '*' && x != corx && y != cory)
        {
            START_GAME_FIELD[x][y] = '*';
            i++;
        }
    }
}
return totalCells - numBombs;
}

int doureal ()
{
    char yn[4];

```

```

while (1)
{
    printf("Do you really want to quit the game? (no/yes): ");
    scanf("%3s", yn);

    if (strcmp(yn, "no") == 0 || strcmp(yn, "No") == 0)
    {
        return 1;
    }

    if (strcmp(yn, "yes") == 0 || strcmp(yn, "Yes") == 0)
    {
        return 2;
    }
    printf("Please enter a valid response.\n\n");
}
}

int confirmInput(int width, int height, const char* message, int defaultWidth, int
defaultHeight)
{
    char ok[10];
    while (1)
    {
        printf("%s", message);
        int c;
        while ((c = getchar()) != '\n' && c != EOF);
        if (fgets(ok, sizeof ok, stdin) == NULL || strcmp(ok, "\n") == 0)
        {
            return 1;
        }
        else if (strcasecmp(ok, "yes\n") == 0 || strcasecmp(ok, "y\n") == 0 ||
strcasecmp(ok, "1\n") == 0)
        {
            return 1;
        }
    }
}

```

```

    else if (strcasecmp(ok, "no\n") == 0 || strcasecmp(ok, "n\n") == 0 ||
strcasecmp(ok, "2\n") == 0)
    {
        return 0;
    }
    else
    {
        printf("Invalid input. Please enter 'yes' or 'no'.\n");
    }
}
return 0;
}

```

```

void dryFd(char **START_GAME_FIELD, char **WORK_FIELD, char
**FLAG_FIELD, int *y, int *x, int row, int col, int corx, int cory, int diff, int dbf,
int *width)
{
    printf("\033[0;0H");
    printf(ColCors"X - coursour;"ColQe" ? - closed cell;"Reset" F - your flag\n");
    printf("Your difficult: ");
    if (diff == 1){printf("easy\n");}
    else if (diff == 2){printf("normal\n");}
    else if (diff == 3){printf("hard\n");}
    else if (diff == 4){printf("very hard\n");}
    else if (diff == 5){printf("IMPOSSIBLE\n");}
    else if (diff == 0){printf("custom\n");}
    printf(ColRed"V sluchae vihoda ili porajeniy ochki ne zachtutsy"Reset);
    for (int n = 0; n < *y; n++)
    {
        printf("\n");
    }
    for (int n = 0; n < *x; n++)
    {
        printf(" ");
    }
}

```

```

printf("┌");
for(int i = 0; i < row; i++){printf("═");}
printf("┐\n");
for(int j = 0; j < col; j++)
{
    for (int n = 0; n < *x; n++)
    {
        printf(" ");
    }
    printf("┆ ");
    for(int i = 0; i < row; i++)
    {
        if(i == cory && j == corx){printf(ColCors "X " Reset);}
        else if (dbf == 1 && START_GAME_FIELD[j][i] == '*'){printf(ColRed"@
"Reset);}
        else if (WORK_FIELD[j][i] != 9){printf(ColNum "%d " Reset,
WORK_FIELD[j][i]);}
        else if (FLAG_FIELD[i][j] == '1'){printf("F ");}
        else{printf(ColQe"? "Reset);}
        fflush(stdout); // очищаем буфер вывода после каждой операции вывода
    }

    printf("┆\n");
    fflush(stdout); // очищаем буфер вывода после каждой операции вывода
}
for (int n = 0; n < *x; n++)
{
    printf(" ");
}
printf("└");
for (int j = 0; j < row; j++){printf("═");}
printf("┘\n");
fflush(stdout); // очищаем буфер вывода после каждой операции вывода
int fdsfdbv = *width-1;
printf("\033[%d;1H", fdsfdbv);
printf("2 - tern down music; Q - quit the game; w/k, a/h, s/j, d/l - move coursour;
c/n - open cells; r - redraw, T - reload game");

```

}

Файл term_tools.c

```
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <string.h>
#include <stdio.h>
#include <termios.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <string.h>

#define MAX_SCORES 10
#define MAX_NAME_LEN 50

void print_leaderboard() {
    FILE *file;
    char line[100];

    file = fopen("scores.txt", "r");
    if (file == NULL) {
        printf("Error: Unable to open file.\n");
        return;
    }

    // Read and print lines from the file
    while (fgets(line, sizeof(line), file) != NULL) {
        if(strcmp(line, "\n") == 0){continue;}
        printf("%s", line);
    }

    fclose(file);
}

void enableRawMode()
{
    struct termios raw;
```

```

    tcgetattr(STDIN_FILENO, &raw);
    raw.c_lflag &= ~(ECHO | ICANON);
    tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);
}

void disableRawMode()
{
    struct termios raw;
    tcgetattr(STDIN_FILENO, &raw);
    raw.c_lflag |= (ECHO | ICANON);
    tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);
}

void term_size(int *height, int *width)//функция для посчета доины размеров терминала
{
    struct winsize size;
    ioctl(STDOUT_FILENO, TIOCGWINSZ, &size);
    *height = size.ws_row; // высота
    *width = size.ws_col; // ширина
}

void indent(int *width, int *height, int *x, int *y, int col, int row)
{
    *x = (*width - (row * 2) - 3)/2;
    *y = (*height - col - 5)/2;
}

void clc__console__(int *dbf)
{
    disableRawMode();
    char command[10];
    printf("\n{clc} Enter command $ ");
    scanf("%s", command);

    if (strcmp(command, "drowb") == 0)

```

```

{
    *dbf = 1;
}
if (strcmp(command, "ndrowb") == 0)
{
    *dbf = 0;
}
if (strcmp(command, "lead") == 0)
{
    system("clear");
    print_leaderboard();
    scanf("%s", command);
}

system("clear");
enableRawMode();
}
void print_leaderboard() {
    FILE *file;
    char line[100];

    file = fopen("scores.txt", "r");
    if (file == NULL) {
        printf("Error: Unable to open file.\n");
        return;
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        if(strcmp(line, "\n") == 0){continue;}
        printf("%s", line);
    }
    fclose(file);
}

void enableRawMode()
{
    struct termios raw;

```

```

    tcgetattr(STDIN_FILENO, &raw);
    raw.c_lflag &= ~(ECHO | ICANON);
    tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);
}

void disableRawMode()
{
    struct termios raw;
    tcgetattr(STDIN_FILENO, &raw);
    raw.c_lflag |= (ECHO | ICANON);
    tcsetattr(STDIN_FILENO, TCSAFLUSH, &raw);
}

void term_size(int *height, int *width)//функция для посчета доины размеров терминала
{
    struct winsize size;
    ioctl(STDOUT_FILENO, TIOCGWINSZ, &size);
    *height = size.ws_row; // высота
    *width = size.ws_col; // ширина
}

void indent(int *width, int *height, int *x, int *y, int col, int row)
{
    *x = (*width - (row * 2) - 3)/2;
    *y = (*height - col - 5)/2;
}

void clc__console__(int *dbf)
{
    disableRawMode();
    char command[10];
    printf("\n{clc} Enter command $ ");
    scanf("%s", command);

    if (strcmp(command, "drowb") == 0)
    {

```

```
    *dbf = 1;
}
if (strcmp(command, "ndrowb") == 0)
{
    *dbf = 0;
}
if (strcmp(command, "lead") == 0)
{
    system("clear");
    print_leaderboard();
    scanf("%s", command);
}

system("clear");
enableRawMode();
}
```

Приложение В

Результат работы программы

Приложение В.1 – Заставка приветствие

