

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ
ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

Журнал практики

Студента **Минеевой Светланы Алексеевны**_____ (ф. и. о.)

Институт **№8 «Компьютерные науки и прикладная математика»**

Кафедра **№805 «Математическая кибернетика»**

Учебная группа **80-105Б-21**

Направление подготовки (специальность) 01.03.04

(шифр)

Прикладная математика

(название направления, специальности)

Вид практики **Ознакомительная**

(учебной, производственной, преддипломной или другой вид практики)

Руководитель практики от МАИ

Кудрявцева Ирина Анатольевна_____

(фамилия, имя, отчество)

(подпись)

_____/_____**Минеева С.А.**_____/ **“12”** июля 2022г.

(подпись студента)

(дата)

1. Место и сроки проведения практики

Сроки проведения практики:

-дата начала практики **29.06.22**

-дата окончания практики **12.07.22**

Наименование предприятия **МАИ**

Название структурного подразделения (отдел, лаборатория)

каф. 805

2. Инструктаж по технике безопасности

_____/_____/ “ 29 ” июня _____ 2022_г.
(подпись проводившего) (дата проведения)

3. Индивидуальное задание студенту

Разработать компьютерную игру «Сапёр» на языке программирования Python, используя модуль Tkinter для создания графического интерфейса. При этом предполагаются следующие правила игры:

Игровое поле разделено на ячейки, некоторые из которых заминированы (количество заминированных ячеек известно). В незаминированных ячейках находятся цифры (от 1 до 8) – это количество мин в соседних ей ячейках. Ячейка может быть и пустой, это значит, что в соседних ей ячейках мин нет. Цель игры – разминировать поле (открыть все ячейки, не содержащие мины, остальные же ячейки пометить флагами). Игрок открывает ячейки, стараясь не открыть ячейку с миной, если же такое случается, то он проигрывает.

При реализации предполагается использование классов. Из правил игры следует, что понадобятся два класса – класс ячеек поля и класс процесса самой игры. Класс ячеек поля создает поле и хранит данные о каждой ячейке. Класс игры реализует сам игровой процесс.

4. План выполнения индивидуального задания

1. Подробно изучить модуль Tkinter на Python
2. Создать основной графический интерфейс для игры
3. Написать код игры
4. Оформить журнал

Руководитель практики от МАИ: Кудрявцева Ирина Анатольевна. _/_____/

Руководитель от предприятия Кудрявцева Ирина Анатольевна _/_____/

_____/_____
(подпись студента)

_____/_____
(дата)

_____/_____
Минеева С.А

_____/_____
“ 29 ” _____ июня _____ 2022 г.

5.Отзыв руководителя практики от предприятия

В процессе выполнения практики студент приобрёл необходимые навыки работы в выбранном языке программирования. Студент успешно использовал принципы объектно-ориентированного программирования, проектирования графического интерфейса и аппарат теории оптимизации.

Материалы, изложенные в отчёте студента, полностью соответствуют индивидуальному заданию

Руководитель от предприятия: Кудрявцева Ирина Анатольевна ____/____/_____
(фамилия, имя, отчество) (подпись)

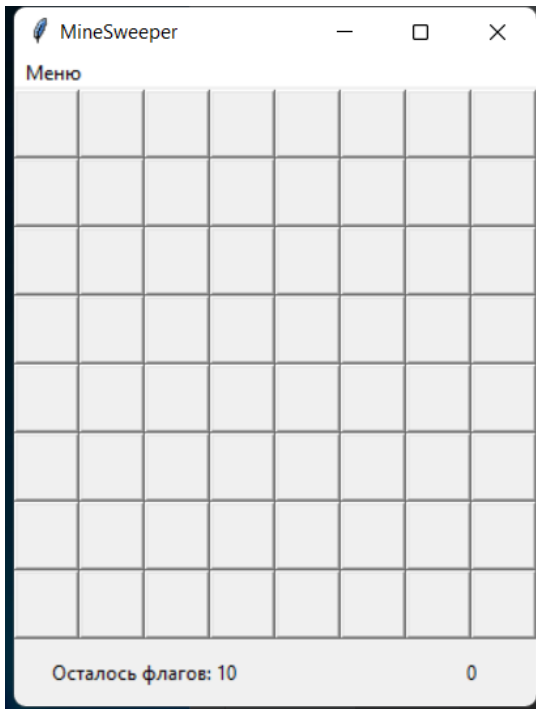
“12” июля 2022 г.

М.П. (печать)

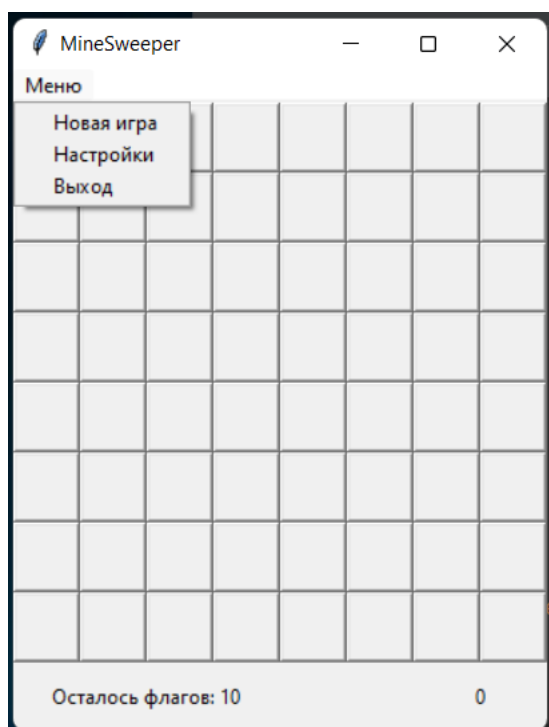
6.Отчет студента о практике

Рассмотрим работу программы:

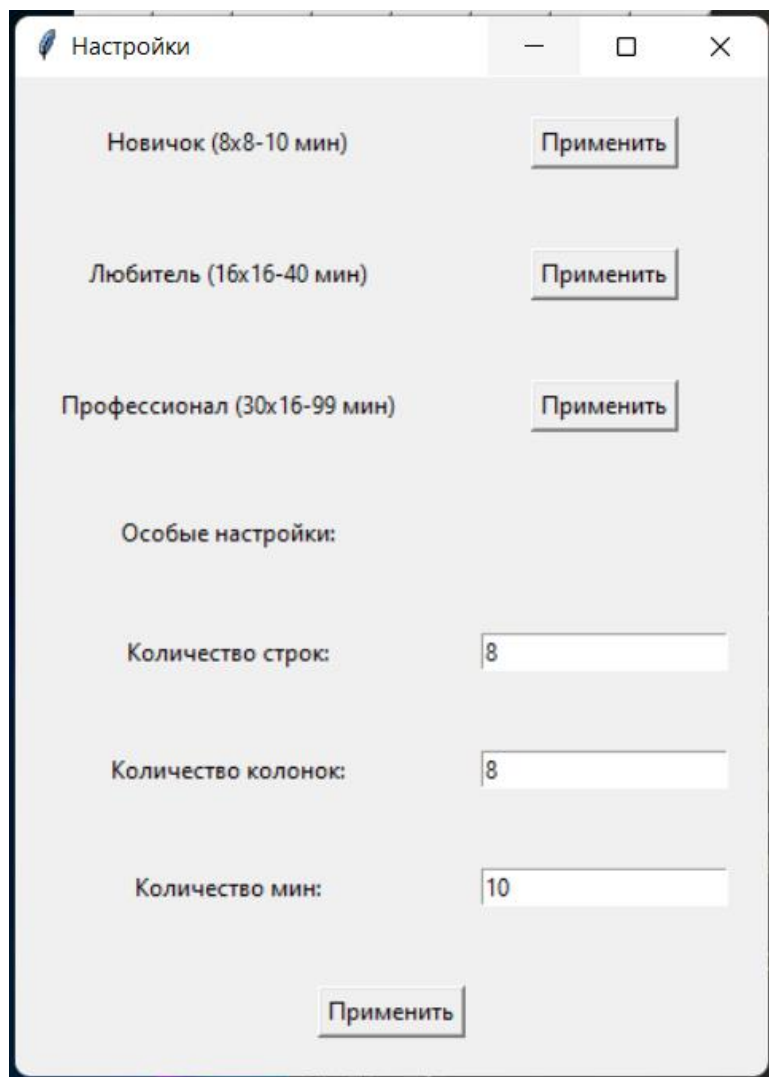
При запуске программы создается поле «Новичок» (8x8 – 10 мин). В левом нижнем углу расположен счетчик того, сколько осталось расставить флагов на поле. В правом нижнем углу расположен таймер, который запускается при первом открытии ячейки на поле. Таймер останавливается при победе или поражении. Окно игры можно растягивать или сужать.



В левом верхнем углу расположено меню. При нажатии на кнопку «меню» открывается выбор:



При выборе кнопки «Новая игра» предыдущий прогресс сбрасывается, начинается новая игра: создается новое поле с параметрами прошлой игры, таймер обнуляется, количество оставшихся флагов вновь становится равным количеству мин на поле. При выборе кнопки «Выход» игрок выходит из игры. При выборе кнопки «Настройки» открывается новое окно для настройки параметров поля:



Настройки

Новичок (8x8-10 мин) Применить

Любитель (16x16-40 мин) Применить

Профессионал (30x16-99 мин) Применить

Особые настройки:

Количество строк: 8

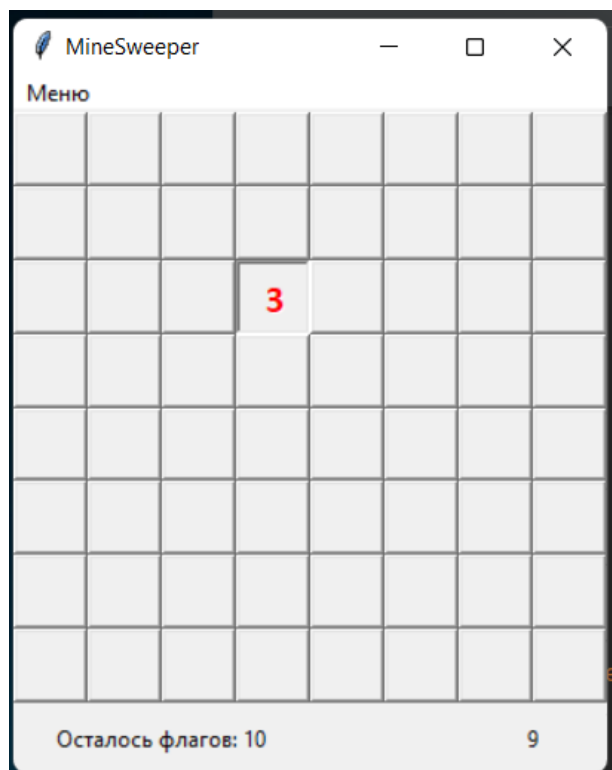
Количество колонок: 8

Количество мин: 10

Применить

Возможен выбор из готовых настроек поля: Новичок (8x8 – 10 мин), Любитель (16x16 – 40 мин), Профессионал (30x16 – 99 мин). Чтобы применить эти параметры, достаточно нажать кнопку «Применить» справа от предложенной корректировки. При её нажатии запуститься новая игра, а поле будет создано по выбранным данным. Возможен выбор особых настроек, то есть количество строк и столбцов поля, количество мин может задать сам игрок. Для этого нужно самому ввести данные и нажать кнопку «Применить», после этого начнется новая игра с выбранными данными. Изначально в этих полях находятся значения параметров, с которыми проходила игра до этого момента.

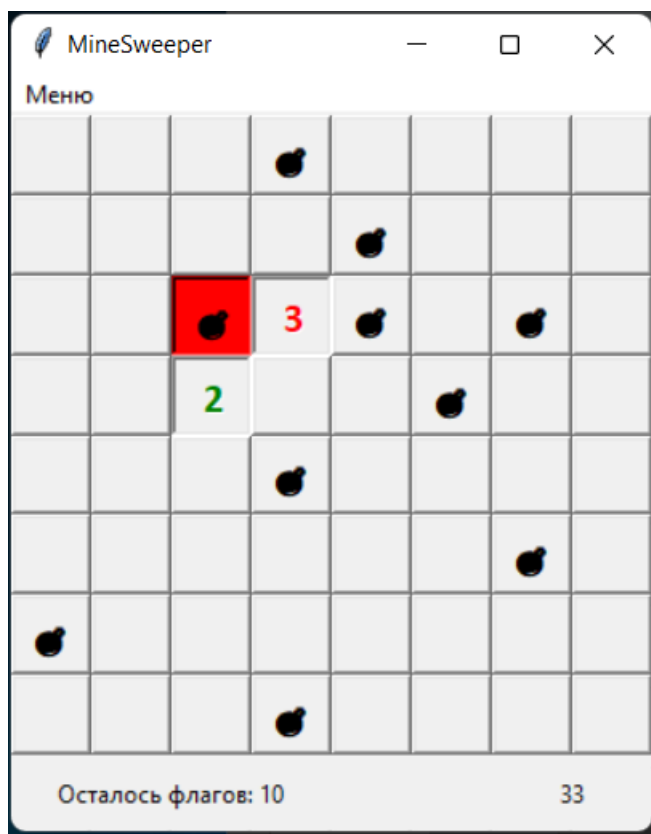
Рассмотрим сам процесс игры. Он начинается при открытии первой ячейки поля, в этот момент запускается таймер. Что важно, проиграть сразу на первом шаге нельзя. Гарантируется, что первая открытая ячейка – не мина, так как мины расставляются уже после её открытия.



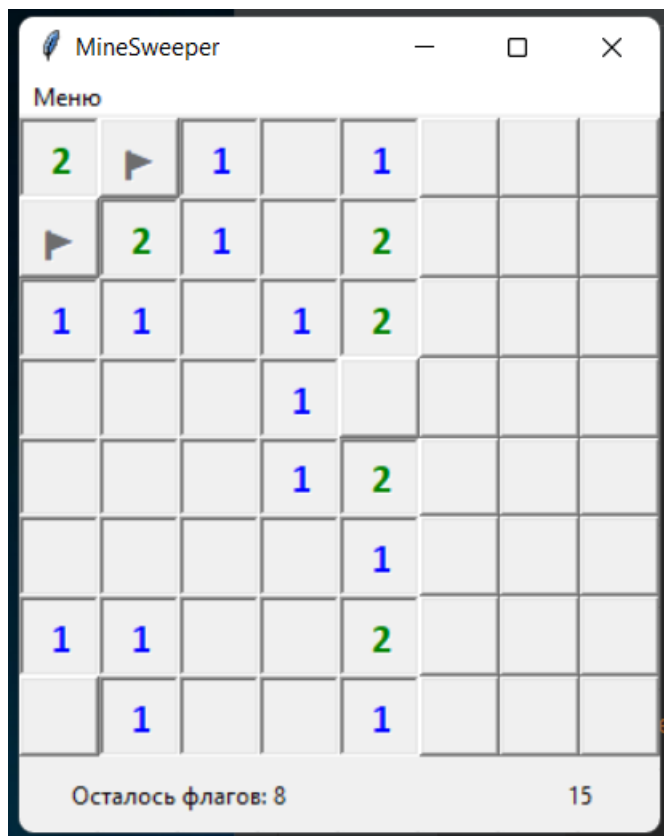
Игрок проигрывает, когда он открывает заминированную ячейку. В таком случае таймер останавливается, открывается окно, оповещающее о проигрыше, а ячейка «взорванной» мины окрашивается в красный цвет.



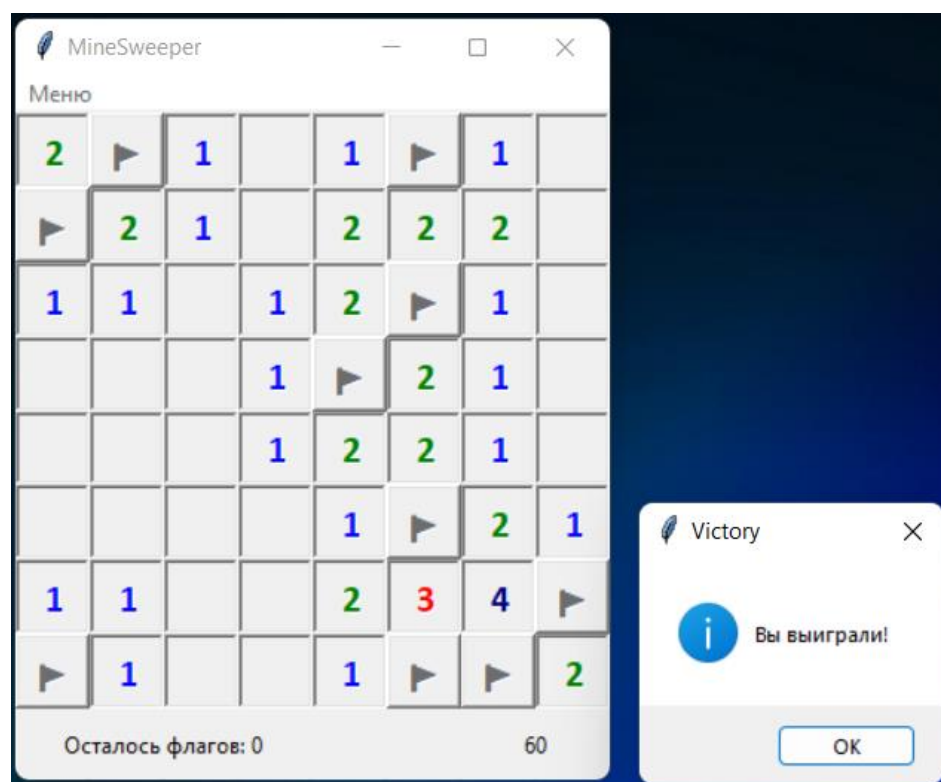
При закрытии окна с оповещением изменяется поле, а именно, игроку показывается расположение оставшихся мин.



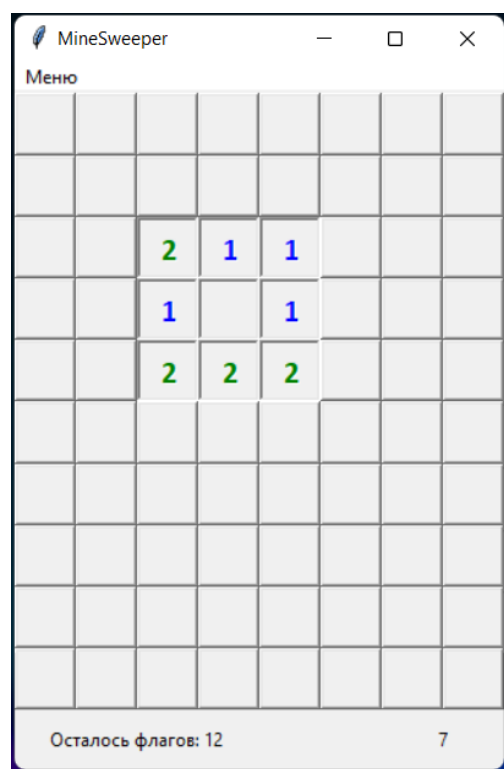
Для победы игрок должен пометить заминированные ячейки флагами. Это можно сделать, кликнув по ячейке правой кнопкой мыши. Поставленный флаг можно и убрать, еще раз нажав по нему правой кнопкой мыши.



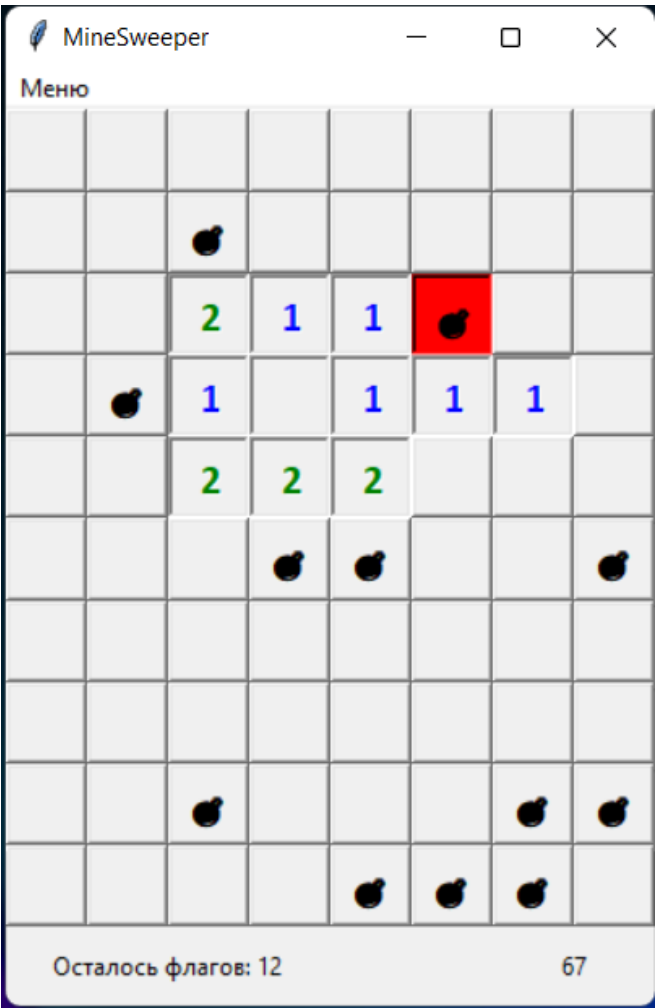
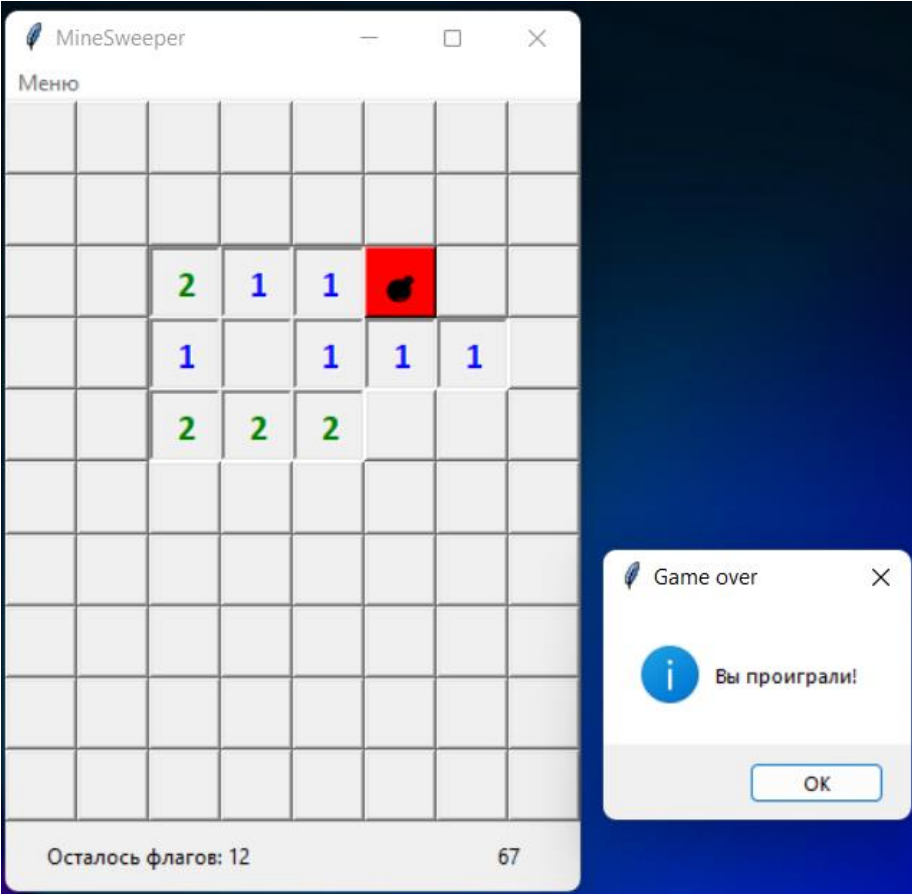
Игрок выиграл, если он верно расставил все флаги и открыл все незаминированные ячейки. В таком случае открывается окно, оповещающее нас о победе.



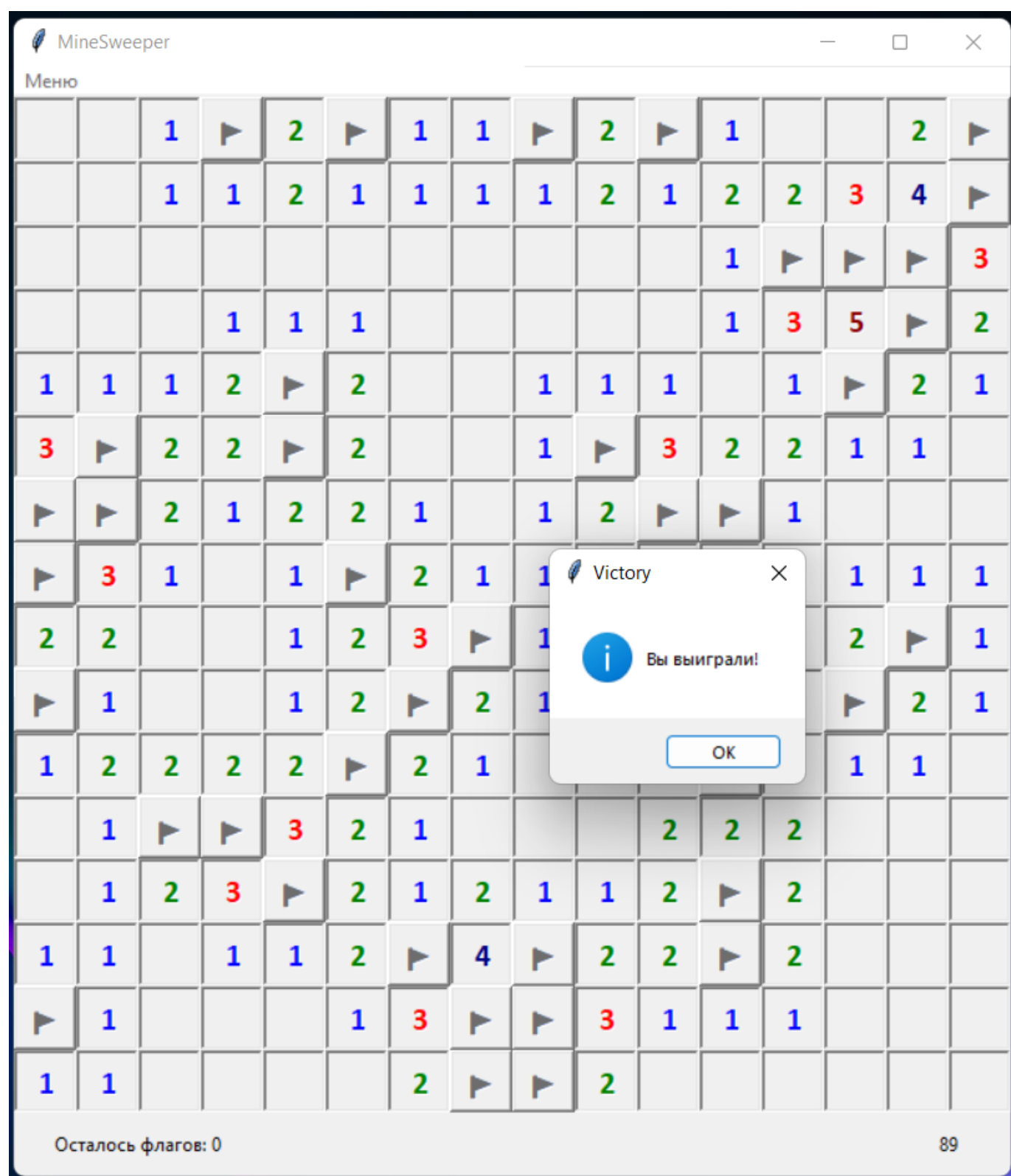
Создадим поле с особыми настройками – поле 10x8 с 12 минами. При первом нажатии на поле увидим интересную вещь – открылось несколько ячеек поля. Данное событие может произойти не только при первом клике, но и при любом. Это результат нажатия на пустую ячейку (вокруг этой ячейки нет мин). В это случае открывается блок незаминированных кнопок при помощи обхода в ширину.



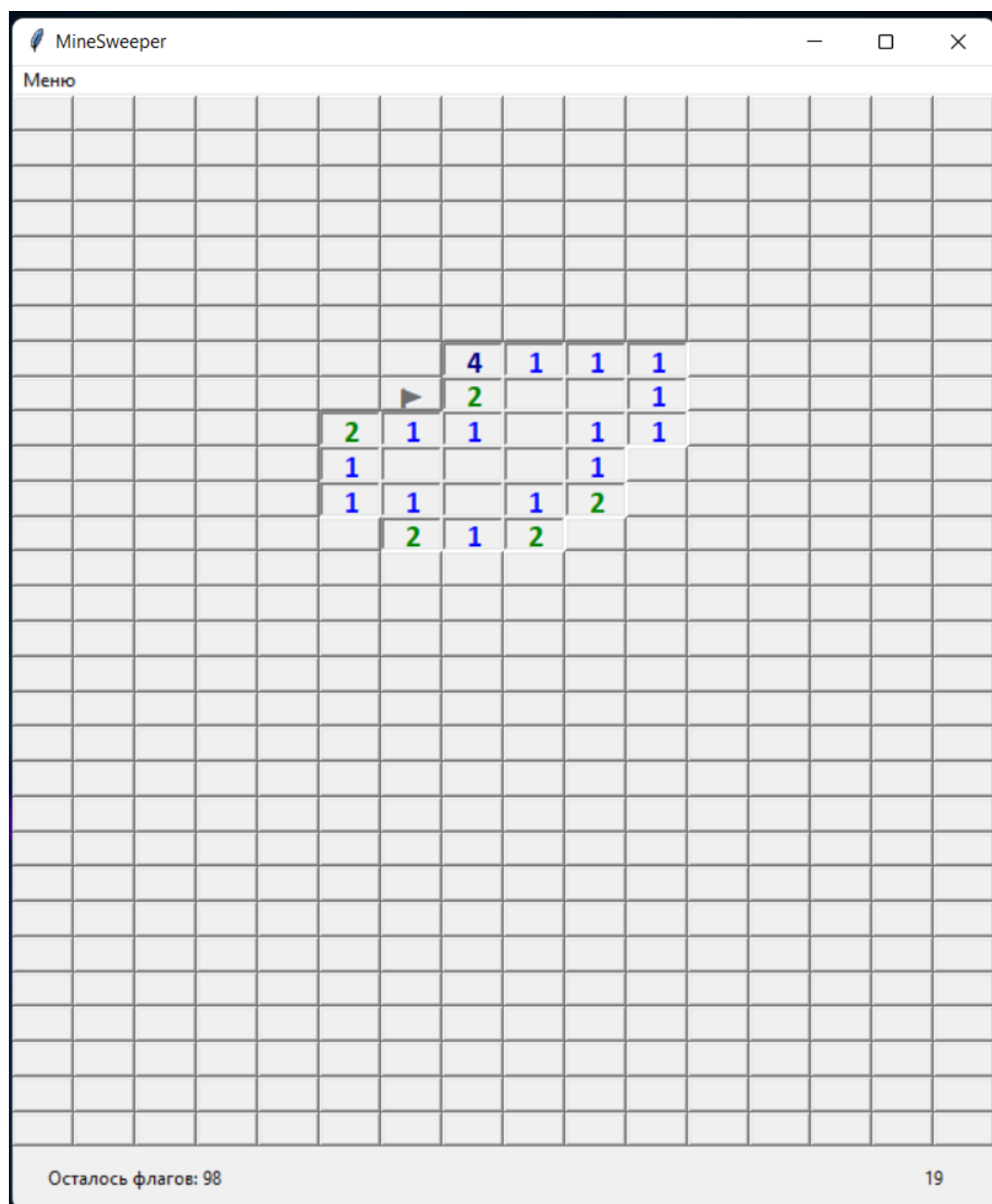
Конец игры из-за проигрыша:

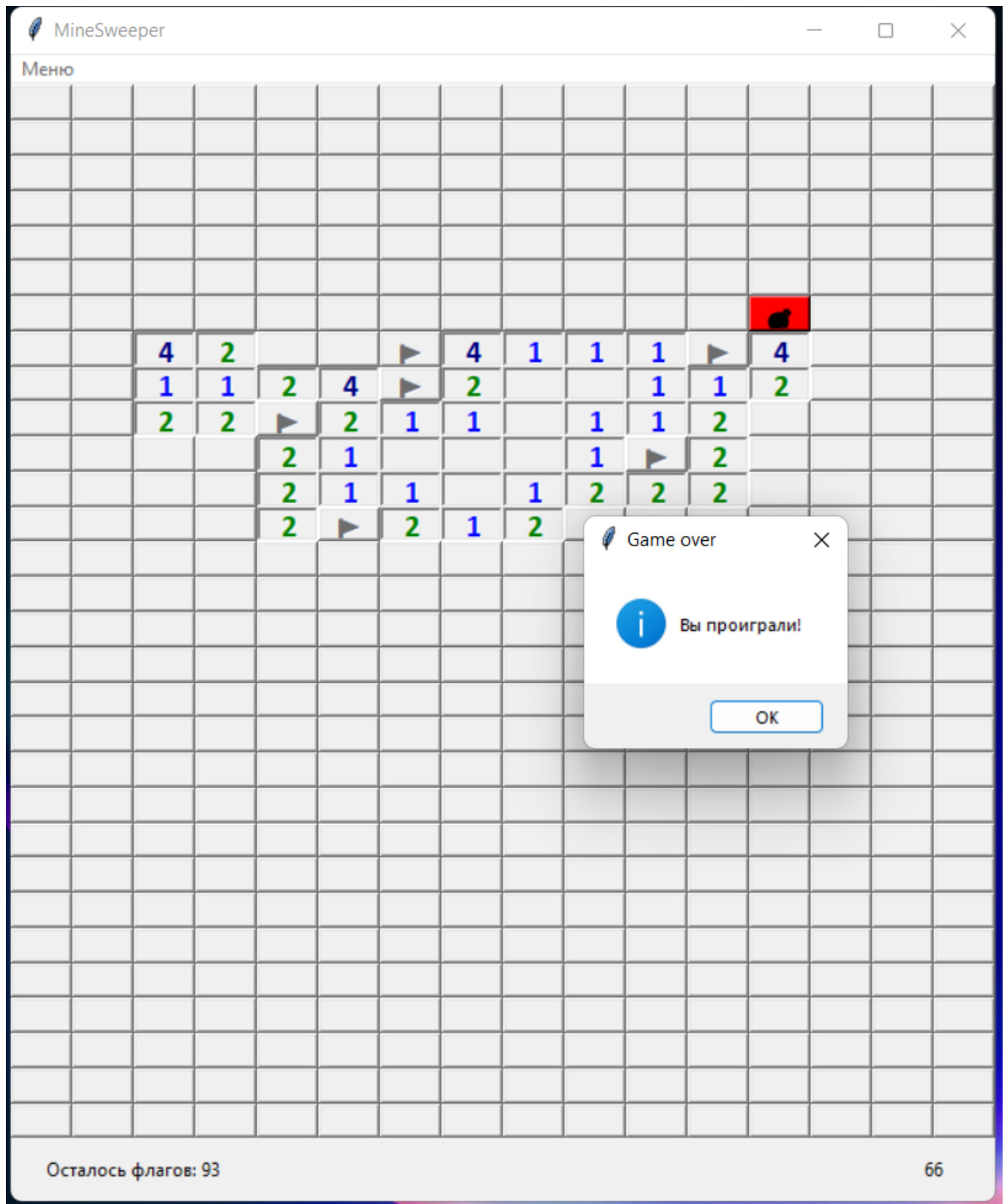


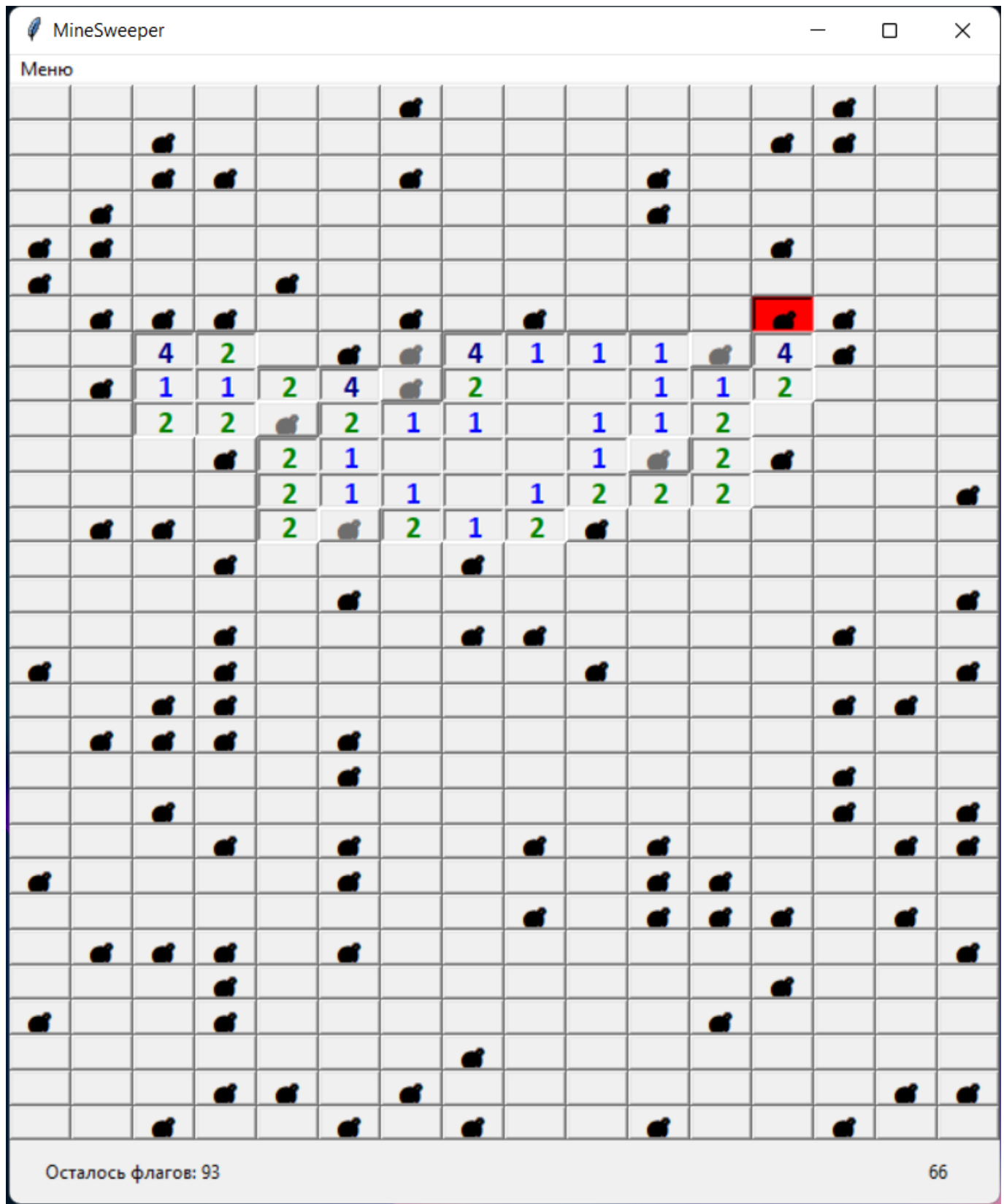
Игра с параметрами «Любитель»:



Игра с параметрами «Профессионал»:







Особенности программной реализации:

- 1) При расставлении цифр (количества мин в соседних ячейках) в ячейки для того чтобы не рассматривать с помощью условного оператора возможные варианты количества

соседних ячеек (3 соседних ячейки для угловой ячейки, 5 соседних для ячеек первых и последних строк и столбцов и 8 соседних в общем случае), созданы барьерные элементы. Проблема просмотра всех вариантов соседних клеток заключается не только в их количестве, но и расположении, например, у угловой верхней правой ячейки соседние ячейки находятся при движении вниз, вправо и по диагонали вниз и вправо, а у угловой левой верхней ячейки при движении вниз, влево и по диагонали влево и вниз. Такая проблема возникает и у ячеек первых и последних столбцов и строк. Барьерные элементы – это дополнительные две строки сверху и снизу поля и два столбца слева и справа поля. Они не отображаются на экране, нужны нам они только для того, чтобы все ячейки рассматривать как ячейки общего вида, то есть у всех ячеек 8 соседних ячеек. Барьерные элементы никак не заполняются, поэтому не повлияют на расчеты.

- 2) В интерфейсе используются различные цвета. Цвета контролируются с помощью словаря, каждому ключу-элементу соответствует название цвета из палитры модуля Tkinter.
- 3) В программе используются принципы объектно-ориентированного программирования. А именно, есть два класса – класс ячеек поля и класс самой игры. Класс ячеек поля создает поле и хранит некоторые данные о каждой ячейке. Класс игры реализует сам игровой процесс.
- 4) Таймер реализуется при помощи библиотек time и threading. Его реализация требует применение последней библиотеки для того, чтобы выполнение функции таймера происходило в потоке GIL, а данные выполненной функции передавались в основную программу. Использование потока необходимо, так как иначе графический интерфейс, созданный в Tkinter, и игра зависнут.
- 5) В графическом интерфейсе игры используются значки мины и флага. Они не являются фотографиями или гифками, они – символы алфавита, поддерживающиеся на каждом устройстве, но немного различающиеся внешним видом.
- 6) При помощи модуля Tkinter, а именно метода bind в программе реализовано различия клика правой и левой кнопкой мыши. При клике левой кнопкой мыши ячейка открывается, а при клике правой кнопкой мыши ставится или убирается флаг.
- 7) Нельзя повторно открыть ячейку, то есть при повторном клике на открытую ячейку ничего не произойдет, никакие программные функции не будут запущены. Это возможно благодаря контролю состоянию кнопки Tkinter (два состояния (state): нормальное (normal) и выключенное (disabled)).

Код:

```
import tkinter as tk
import time
from threading import Thread
from random import shuffle
from tkinter.messagebox import showinfo, showerror

colors = {1: 'blue', 2: 'green', 3: 'red', 4: 'navy', 5: 'darkred', 6: 'magenta', 7: 'purple', 8: 'gold',}

class MyButton(tk.Button):

    def __init__(self, master, x, y, number=0, *args, **kwargs):
        super(MyButton, self).__init__(master, font='Calibri 15 bold', width=3, *args, **kwargs)
        self.x, self.y, self.number, self.count_bomb = x, y, number, 0
        self.is_mine, self.is_open = False, False

    def __repr__(self):
        return f'MyButton {self.x} {self.y} {self.number} {self.is_mine}'

class MineSweeper:
    window = tk.Tk()
    window.title("MineSweeper")
    row, column, mines, flags, clock, kol = 8, 8, 10, 0, 0, 0
    c_m = mines
    indexes_mines, indexes_flags = list(), list()
    is_game_over, is_victory, is_first_click, running = False, False, True, True

    def __init__(self):
        self.buttons = []
        for i in range(MineSweeper.row + 2):
            temp = []
            for j in range(MineSweeper.column + 2):
                btn = MyButton(MineSweeper.window, x=i, y=j)
                btn.config(command=lambda button=btn: self.click(button))
                btn.bind("<Button-3>", self.right_click)
                temp.append(btn)
            self.buttons.append(temp)
        self.flags_remain = tk.Label(MineSweeper.window, text='Осталось флагов: ' +
str(MineSweeper.c_m))
        self.flags_remain.grid(row=MineSweeper.row + 2, column=1, columnspan=4, padx=10,
pady=10)
        self.time = tk.Label(MineSweeper.window, text='0')
        self.time.grid(row=MineSweeper.row + 2, column=MineSweeper.column - 1, columnspan=2,
padx=10, pady=10)

    def right_click(self, event):
        if MineSweeper.is_game_over: return
        cur_btn = event.widget
        if cur_btn['state'] == 'normal':
```

```

cur_btn['state'] = 'disabled'
cur_btn['text'] = '🚩'
MineSweeper.flags += 1
MineSweeper.kol += 1
MineSweeper.indexes_flags.append(cur_btn.number)
MineSweeper.c_m -= 1
self.flags_remain.config(text='Осталось флагов: ' + str(MineSweeper.c_m))
if MineSweeper.flags == MineSweeper.mines and MineSweeper.kol ==
MineSweeper.column * MineSweeper.row:
    MineSweeper.indexes_mines.sort()
    print(MineSweeper.indexes_mines)
    MineSweeper.indexes_flags.sort()
    print(MineSweeper.indexes_flags)
    if MineSweeper.indexes_mines == MineSweeper.indexes_flags:
        MineSweeper.is_victory, MineSweeper.running = True, False
        showinfo('Victory', 'Вы выиграли!')
elif cur_btn['text'] == '🚩':
    cur_btn['text'] = ''
    cur_btn['state'] = 'normal'
    MineSweeper.flags -= 1
    MineSweeper.kol -= 1
    MineSweeper.indexes_flags.remove(cur_btn.number)
    MineSweeper.c_m += 1
    self.flags_remain.config(text='Осталось флагов: ' + str(MineSweeper.c_m))

def click(self, clicked_button: MyButton):
    if MineSweeper.is_game_over: return
    if MineSweeper.is_victory: return
    MineSweeper.kol += 1
    if MineSweeper.is_first_click == True:
        MineSweeper.is_first_click = False
        self.insert_mines(clicked_button.number)
        self.count_mines_in_buttons()
        self.print_buttons()
        Thread(target=self.autoc).start()
    color = colors.get(clicked_button.count_bomb, 'black')
    if clicked_button.is_mine:
        clicked_button.config(text="💣", background='red', disabledforeground='black')
        clicked_button.is_open, MineSweeper.is_game_over, MineSweeper.running = True, True,
False
        showinfo('Game over', 'Вы проиграли!')
        for i in range(1, MineSweeper.row + 1):
            for j in range(1, MineSweeper.column + 1):
                btn = self.buttons[i][j]
                if btn.is_mine: btn['text'] = '💣'
    elif clicked_button.count_bomb:
        clicked_button.config(text=clicked_button.count_bomb, disabledforeground=color)
        clicked_button.is_open = True
    else:

```

```

        clicked_button.config(text="", disabledforeground=color)
        clicked_button.is_open = True
        for i in [-1, 0, 1]:
            for j in [-1, 0, 1]:
                btn = self.buttons[clicked_button.x + i][clicked_button.y + j]
                if not btn.is_open and btn.number != 0:
                    self.click(btn)
            if Minesweeper.flags == Minesweeper.mines and Minesweeper.kol == Minesweeper.column *
Minesweeper.row:
                Minesweeper.indexes_mines.sort()
                print(Minesweeper.indexes_mines)
                Minesweeper.indexes_flags.sort()
                print(Minesweeper.indexes_flags)
                if Minesweeper.indexes_mines == Minesweeper.indexes_flags:
                    Minesweeper.is_victory, Minesweeper.running = True, False
                    showinfo('Victory', 'Вы выиграли!')
                clicked_button.config(state='disable')
                clicked_button.config(relief='sunken')

def reload(self):
    [child.destroy() for child in self.window.winfo_children()]
    Minesweeper.c_m = Minesweeper.mines
    Minesweeper.is_first_click = True
    Minesweeper.indexes_mines, Minesweeper.indexes_flags = list(), list()
    Minesweeper.is_victory, Minesweeper.is_game_over, Minesweeper.running = False, False,
True
    Minesweeper.clock, Minesweeper.flags, Minesweeper.kol = 0, 0, 0
    self.__init__()
    self.create_widgets()

def create_setting_win(self):
    win_setting = tk.Toplevel(self.window)
    win_setting.wm_title('Настройки')
    tk.Label(win_setting, text='Новичок (8x8-10 мин)').grid(row=0, column=0, padx=20, pady=20)
    nov_btn = tk.Button(win_setting, text="Применить",
                        command=lambda: self.nov())
    nov_btn.grid(row=0, column=1, columnspan=2, padx=20, pady=20)
    tk.Label(win_setting, text='Любитель (16x16-40 мин)').grid(row=1, column=0, padx=20,
pady=20)
    lub_btn = tk.Button(win_setting, text="Применить",
                        command=lambda: self.lub())
    lub_btn.grid(row=1, column=1, columnspan=2, padx=20, pady=20)
    tk.Label(win_setting, text='Профессионал (30x16-99 мин)').grid(row=2, column=0, padx=20,
pady=20)
    pro_btn = tk.Button(win_setting, text="Применить",
                        command=lambda: self.pro())
    pro_btn.grid(row=2, column=1, columnspan=2, padx=20, pady=20)
    tk.Label(win_setting, text='Особые настройки:').grid(row=3, column=0, padx=20, pady=20)
    tk.Label(win_setting, text='Количество строк:').grid(row=4, column=0)

```

```

tk.Label(win_setting, text='Количество колонок:').grid(row=5, column=0)
tk.Label(win_setting, text='Количество мин:').grid(row=6, column=0)
row_entry = tk.Entry(win_setting)
row_entry.insert(0, Minesweeper.row)
row_entry.grid(row=4, column=1, padx=20, pady=20)
column_entry = tk.Entry(win_setting)
column_entry.insert(0, Minesweeper.column)
column_entry.grid(row=5, column=1, padx=20, pady=20)
mines_entry = tk.Entry(win_setting)
mines_entry.insert(0, Minesweeper.mines)
mines_entry.grid(row=6, column=1, padx=20, pady=20)
save_btn = tk.Button(win_setting, text="Применить",
                      command=lambda: self.change_settings(row_entry, column_entry, mines_entry))
save_btn.grid(row=7, column=0, columnspan=2, padx=20, pady=20)

def change_settings(self, row: tk.Entry, column: tk.Entry, mines: tk.Entry):
    try:
        int(row.get()), int(column.get()), int(mines.get())
    except ValueError:
        showerror('Ошибка', 'Вы ввели неправильное значение!')
    return
    Minesweeper.row, Minesweeper.column, Minesweeper.mines = int(row.get()),
int(column.get()), int(mines.get())
    self.reload()

def nov(self):
    Minesweeper.row, Minesweeper.column, Minesweeper.mines = 8, 8, 10
    self.reload()

def lub(self):
    Minesweeper.row, Minesweeper.column, Minesweeper.mines = 16, 16, 40
    self.reload()

def pro(self):
    Minesweeper.row, Minesweeper.column, Minesweeper.mines = 30, 16, 99
    self.reload()

def create_widgets(self):
    menubar = tk.Menu(self.window)
    self.window.config(menu=menubar)
    settings_menu = tk.Menu(menubar, tearoff=0)
    settings_menu.add_command(label='Новая игра', command=self.reload)
    settings_menu.add_command(label='Настройки', command=self.create_setting_win)
    settings_menu.add_command(label='Выход', command=self.window.destroy)
    menubar.add_cascade(label='Меню', menu=settings_menu)
    count = 1
    for i in range(1, Minesweeper.row + 1):
        for j in range(1, Minesweeper.column + 1):
            btn = self.buttons[i][j]

```

```

        btn.number = count
        btn.grid(row=i, column=j, sticky='NWES')
        count += 1
    for i in range(1, Minesweeper.row + 1):
        tk.Grid.rowconfigure(self.window, i, weight=1)
    for j in range(1, Minesweeper.column + 1):
        tk.Grid.columnconfigure(self.window, i, weight=1)

def open_all_buttons(self):
    for i in range(Minesweeper.row + 2):
        for j in range(Minesweeper.column + 2):
            btn = self.buttons[i][j]
            if btn.is_mine:
                btn.config(text="💣", background='red', disabledforeground='black')
            elif btn.count_bomb in colors:
                color = colors.get(btn.count_bomb, 'black')
                btn.config(text=btn.count_bomb, fg=color)

def print_buttons(self):
    for i in range(1, Minesweeper.row + 1):
        for j in range(1, Minesweeper.column + 1):
            btn = self.buttons[i][j]
            if btn.is_mine:
                print('B')
            else:
                print(btn.count_bomb)

def insert_mines(self, number: int):
    index_mines = self.get_mines_places(number)
    print(index_mines)
    for i in range(1, Minesweeper.row + 1):
        for j in range(1, Minesweeper.column + 1):
            btn = self.buttons[i][j]
            if btn.number in index_mines:
                btn.is_mine = True

def count_mines_in_buttons(self):
    for i in range(1, Minesweeper.row + 1):
        for j in range(1, Minesweeper.column + 1):
            btn = self.buttons[i][j]
            count_bomb = 0
            if not btn.is_mine:
                for row_dx in [-1, 0, 1]:
                    for col_dx in [-1, 0, 1]:
                        neighbour = self.buttons[i + row_dx][j + col_dx]
                        if neighbour.is_mine:
                            count_bomb += 1
            btn.count_bomb = count_bomb

```

```

def get_mines_places(self, exclude_number: int):
    indexes = list(range(1, Minesweeper.column * Minesweeper.row + 1))
    indexes.remove(exclude_number)
    shuffle(indexes)
    Minesweeper.indexes_mines = indexes[:Minesweeper.mines]
    return indexes[:Minesweeper.mines]

def start(self):
    self.create_widgets()
    Minesweeper.window.mainloop()

def autoc(self):
    while Minesweeper.running:
        time.sleep(1)
        Minesweeper.clock += 1
        self.time.config(text=int(Minesweeper.clock))

game = Minesweeper()
game.start()

```

Список литературы:

- 1) Майкл Доусон «Программируем на Python»
- 2) Марк Лутц «Программирование на Python»
- 3) Алан Мур «Mastering GUI Programming with Python»
- 4) Микки Нардо «Учебное пособие по Tkinter для новичков»
<https://www.russianlutheran.org/python/nardo/nardo.html>
- 5) О Tkinter: <https://younglinux.info/tkinter/tkinter>