МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные технологии и общенаучные дисциплины»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**на лабораторную работу**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Разработка компьютерной игры «крестики-нолики»**

Р.02069337. №23/36-9 ТЗ-00

Листов 19

|  |
| --- |
|  |
| Руководитель разработки: |
| кандидат технических наук, доцент |
| Шишкин Вадим Викторинович |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |
|  |
| Исполнитель: |
| студент гр. АИСТбд-21 |
| Ковалев Никита Александрович |
| «\_\_\_» \_\_\_\_\_\_\_\_ 202 г. |

2024

**Содержание**

|  |  |
| --- | --- |
| Аннотация................................................................................................................. | 3 |
| Техническое задание................................................................................................ | 5 |
| Пояснительная записка............................................................................................ | 9 |
| Руководство программиста...................................................................................... | 12 |
| Текст программы...................................................................................................... | 14 |
|  |  |
|  |  |

**Аннотация**

В этом лабораторном отчете подробно описана реализация игры «Крестики-нолики» на языке Python с искусственным интеллектом, использующим Tkinter для графического пользовательского интерфейса и алгоритм Minimax с альфа-бета-отсечением для принятия решений ИИ. В отчете рассматривается реализация основной игровой логики, стратегии искусственного интеллекта и взаимодействия с пользователем.

**Текст программы**

|  |
| --- |
| import tkinter as tk from tkinter import messagebox   class TicTacToe:  def \_\_init\_\_(self, root):  self.root = root  self.root.title("Крестики-Нолики (игра против компьютера)")   self.current\_player = "X" # Игрок  self.computer\_player = "O" # Компьютер  self.board = [""] \* 9  self.buttons = []   self.start\_screen()   def start\_screen(self):  self.clear\_window()  label = tk.Label(self.root, text="Кто ходит первым?", font=("Arial", 18))  label.pack(pady=20)   first\_player\_frame = tk.Frame(self.root)  first\_player\_frame.pack()   player\_btn = tk.Button(first\_player\_frame, text="Я", command=self.player\_starts)  player\_btn.grid(row=0, column=0, padx=10)   computer\_btn = tk.Button(first\_player\_frame, text="Компьютер", command=self.computer\_starts)  computer\_btn.grid(row=0, column=1, padx=10)   def clear\_window(self):  for widget in self.root.winfo\_children():  widget.destroy()   def player\_starts(self):  self.current\_player = "X"  self.computer\_player = "O"  self.start\_game()   def computer\_starts(self):  self.current\_player = "O"  self.computer\_player = "X"  self.start\_game()  self.computer\_move()   def start\_game(self):  self.board = [""] \* 9  self.clear\_window()  self.create\_buttons()  self.reset\_button()   def create\_buttons(self):  # Очистка старых кнопок  self.buttons = []   for i in range(9):  button = tk.Button(self.root, text="", font=("Arial", 24), width=5, height=2,  command=lambda i=i: self.on\_player\_click(i))  button.grid(row=i // 3, column=i % 3)  self.buttons.append(button)   def on\_player\_click(self, index):  if self.board[index] == "" and self.check\_winner() is None:  self.board[index] = self.current\_player  self.buttons[index].config(text=self.current\_player)   winner = self.check\_winner()  if winner:  messagebox.showinfo("Игра окончена!", f"Победитель: {winner}")  self.reset\_game()  elif "" not in self.board:  messagebox.showinfo("Игра окончена!", "Ничья!")  self.reset\_game()  else:  self.computer\_move()   def computer\_move(self):  best\_score = -float('inf')  best\_index = None   for i in range(9):  if self.board[i] == "":  self.board[i] = self.computer\_player  score = self.minimax(self.board, False)  self.board[i] = ""   if score > best\_score:  best\_score = score  best\_index = i   self.board[best\_index] = self.computer\_player  self.buttons[best\_index].config(text=self.computer\_player)   winner = self.check\_winner()  if winner:  messagebox.showinfo("Игра окончена!", f"Победитель: {winner}")  self.reset\_game()  elif "" not in self.board:  messagebox.showinfo("Игра окончена!", "Ничья!")  self.reset\_game()   def minimax(self, board, is\_maximizing):  winner = self.check\_winner()  if winner == self.computer\_player:  return 1  elif winner == self.current\_player:  return -1  elif "" not in board:  return 0   if is\_maximizing:  best\_score = -float('inf')  for i in range(9):  if board[i] == "":  board[i] = self.computer\_player  score = self.minimax(board, False)  board[i] = ""  best\_score = max(score, best\_score)  return best\_score  else:  best\_score = float('inf')  for i in range(9):  if board[i] == "":  board[i] = self.current\_player  score = self.minimax(board, True)  board[i] = ""  best\_score = min(score, best\_score)  return best\_score   def check\_winner(self):  for i in range(3):  # вертикали  if self.board[i \* 3] == self.board[i \* 3 + 1] == self.board[i \* 3 + 2] != "":  return self.board[i \* 3]  # горизонты  if self.board[i] == self.board[i + 3] == self.board[i + 6] != "":  return self.board[i]   # диагонали  if self.board[0] == self.board[4] == self.board[8] != "":  return self.board[0]  if self.board[2] == self.board[4] == self.board[6] != "":  return self.board[2]   return None   def reset\_game(self):  self.board = [""] \* 9  for button in self.buttons:  button.config(text="")  self.start\_screen()   def reset\_button(self):  reset\_btn = tk.Button(self.root, text="Сбросить игру", command=self.reset\_game)  reset\_btn.grid(row=3, column=0, columnspan=3)  root = tk.Tk() game = TicTacToe(root) root.mainloop() |