ВВЕДЕНИЕ

1.1. Наименование продукта

Интеллектуальная логическая игра «Крестики-нолики».

1.2. Краткая характеристика области применения

Игра «Крестики-нолики» позволяет пользователю выбрать поле 9х9, 15х15, 45х45, играть против компьютера и ходить крестиками во время игры. Данная игра предназначена для тренировки логического мышления пользователя.

1. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1. Документ, на основании которого ведётся разработка

Задание на курсовую работу по дисциплине «Программирование на языке высокого уровня».

2.2. Организация, утвердившая документ

Национальный Исследовательский  Томский политехнический университет.

1. НАЗНАЧЕНИЕ РАЗРАБОТКИ

Данная разработка дает возможность пользователю развить его логическое мышление, выбор поля позволяет заинтересовать пользователя и делает процесс игры разнообразным. Игра повышает интеллектуальный уровень пользователя, развивает его внимание и рефлекторные способности.

1. ТРЕБОВАНИЯ К РАЗРАБОТКЕ

4.1.Требования к функциональным характеристикам

4.1.1. Программа должна содержать игровое поле

     4.1.2.  Программа должна содержать главное меню, состоящее из четырех разделов «Begin», «Record», «About», «Exit», и каждый раздел меню должен иметь подменю. Первые три  раздела должны содержать кнопку «Exit» для возврата в главное меню.

   4.1.3.  Программа должна иметь простой графический интерфейс.

   4.1.4. Программа должна предоставить возможность осуществление управлением игрой с помощью манипулятора - мышь.

   4.1.5. Программа должна предоставить возможность пользователю при его заходе  в раздел «Begin», выбрать одно из трех возможных полей: 9x 9, 15x 15, 45x 45.

   4.1.6. Программа должна предоставить возможность пользователю в ходе игрового процесса наблюдать за своим количеством побед, за количеством побед компьютера, видеть текущие победы в специальном окне  левом верхнем углу.

             4.1.7. Программа должна предоставить возможность пользователю в ходе игрового процесса в случае выигрыша или проигрыша видеть в левом верхнем углу сообщение о победе или неудаче: «Youwin!»  или  «Youlose!»

             4.1.8. Программа должна содержать в игровом поле три кнопки «Start», «Save», «Exit». Первая кнопка должна позволить пользователю начать игру, вторая – сохранить своё текущее количество побед, а третья – выйти из игры .

 4.1.9. Программа должна содержать раздел главного меню «Record», где пользователю должна предоставляться возможность увидеть историю побед .

           4.1.10. Программа должна содержать раздел главного меню «About»,  где пользователю должна предоставляться возможность прочитать краткую информацию об игре и правила игры.

 4.1.11. Программа должна предоставлять возможность игроку всегда ходить первым и играть фишкой – «крестик», а компьютер должен всегда ходить фишкой – «нолик».

           4.1.12. Программа должна содержать стратегию ходов компьютера, которая должна быть  основана на оценочной функции. Каждая клетка игровой сетки должна иметь свой вес. И где вес максимальный, туда компьютер и должен ходить, причем ставить он должен последующую фишку рядом со своей стоящей фишкой, таким образом, выстраивая возможный выигрышный ряд. Но также, он должен блокировать возможный выигрышный ряд игрока.

4.1.13. Программа должна поставить задачу, как игроку, так и компьютеру выстроить свой ряд из пяти фишек или по горизонтали, или по вертикали, или по диагоналям.

4.2.Требованияк надёжности

4.2.1. Программа должна отслеживать существование файлов «data1.txt» и «data.txt» при их открытии, в противном случае выдать ошибку.

4.2.2. Программа при сохранении в файл «data.txt» должна проверить существование файла с таким же именем, при наличии такого файла предложить перезапись уже существующего файла.

4.2.3. Программа должна корректно отображать данные в английской кодировке.

4.2.4. Программа должна осуществлять проверку на инициализацию графического режима, в случае ошибки выводить сообщение на экран.

4.3. Требования к составу и параметрам технических средств

Для функционирования игры «Крестики-нолики» необходим компьютер IBMPC совместимый и следующие технические средства:

процессорIntel 80286 или совместимый;

объем свободной оперативной памяти 700 Кб;

объем необходимой памяти на жестком диске 600 Кб;

стандартныйVGA-монитор или совместимый;

стандартная клавиатура;

манипулятор «мышь»;

4.4. Требования к информационной и программной совместимости

Для функционирования текстового редактора необходимо наличие операционной системыMS-DOS или совместимой. Язык интерфейса – английский.

1. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Предварительный состав программной документации:

«Техническое задание», текст программы с комментариями, описание алгоритма и программы.

«Руководство пользователя».

1. ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Существует эмулятор DosBox, который позволяет увеличить быстродействие выполнение программы и отрисовку графики, что значительно экономит время  при выполнении программы.

7. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

ТЗ - техническое задание, ПЗ -  пояснительная записка.

ВВЕДЕНИЕ

Данная курсовая работа посвящена разработке игры «Крестики-нолики»,реализованная с применением объектно-ориентированного программирования. Курсовая работа ставит перед собой цель проработки и упорядочивания знаний, полученных в течение двух семестров изучения дисциплины «Программирование на языке высокого уровня». Курсовая работа представляет собой индивидуальное задание повышенной сложности по разработке программы на изучаемом  языке высокого уровня.

В курсовой работе будут рассмотрены: алгоритм выполнения  стратегии ходов компьютера, работа с файлами, графический режим, объектно-ориентированное программирование. Все элементы будут рассмотрены на языке программирования  высокого уровня Python.

Выполнение задания требует комплексного применения своих знаний как в части владения языком программирования, так и в части умения создавать и реализовывать сложные алгоритмы.

Игра «Крестики-нолики», описанная в данной работе, является полноценным программным продуктом, готовым для использования. Она обладает доступным, понятным и наглядным для пользователя интерфейсом.

ПОСТАНОВКА ЗАДАЧИ

Используя ООП, написать программу, реализующую идею игры. Выполнить переопределение стандартных операций для абстрактных типов данных. Использовать конструкторы, деструкторы. Использовать графический режим.

ВЫБОР И ПРОРАБОТКА ПРОГРАММНЫХ СРЕДСТВ

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

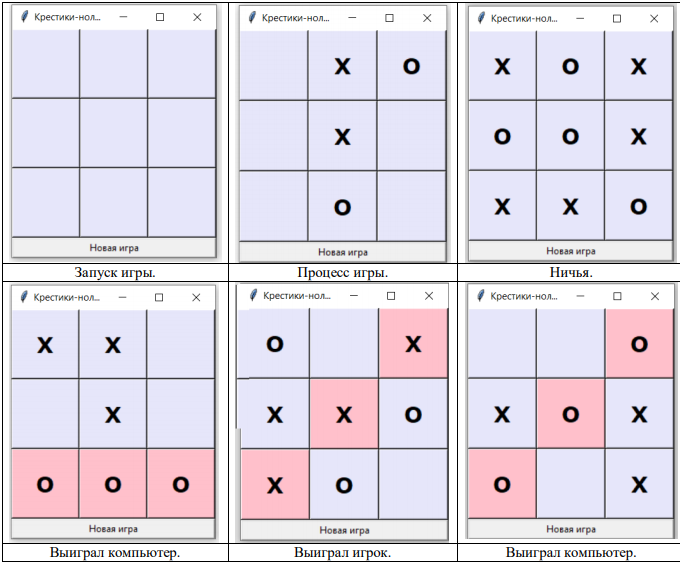
Объектно-ориентированное программирование — это обобщение структурного и модульного программирования для оптимизации создания программ и программных комплексов. ООП позволяет быстро и с минимальным количеством ошибок написать программу, код которой легко понять другим и самому, и, следовательно, изменить при необходимости. Язык программирования является объектно-ориентированным, когда выполняются условия:

* поддерживаются объекты или абстракции данных, имеющие интерфейс в виде именованных операций и собственные данные с ограничением доступа к ним;
* объекты относятся к соответствующим типам;
* типы (классы) могут наследовать атрибуты супертипов (базовых классов);

Три основные положения объектно-ориентированного программирования:

* инкапсуляция
* наследование
* полиморфизм

ПРОЕКТИРОВАНИЕ ПРОГРАММЫ



Внизу расположена кнопка «Новая игра» для начала новой игры. По её нажатию игровое поле очистится и можно будет начать игру заново. Игровое поле представляет собой матрицу 3 х 3, т.е. три строки (row) и три столбца (col). Для отображения графики будем использовать стандартную библиотеку Tkinter. Так же нам потребуется библиотека random для того, чтобы получать случайные числа, благодаря которым ходы компьютера будут неожиданными. Подключаем их с помощью import. Создаем окно root, устанавливаем ему заголовок и объявляем необходимые переменные:

game\_run – в эту переменную будем записывать False при завершении игры, чтобы запретить делать ходы когда уже выявлен победитель.

field – это будет двумерный список, в котором будут храниться кнопки игрового поля. Ходом будет изменение надписи на кнопке на символ «X» или «O».

cross\_count в этой переменной мы будем отслеживать количество крестиков на поле. Чтобы по выставлению пятого крестика, в случае если никто не выиграл фиксировать ничью.

from tkinter import \*

import random root = Tk()

root.title('Крестики-нолики')

game\_run = True field = [] cross\_count = 0

Обработка нажатия кнопок

Для этого используются две функции:

* 1. Функция new\_game() будет вызываться при нажатии кнопки начала новой игры. Функция выполняет следующие действия:
  2. • На поле убираются все крестики и нолики. Цвет кнопок делаем бледно-лиловым (lavender). Для этого используются циклы for по строкам (row) и по столбцам (col).
  3. • Устанавливаем глобальные переменные game\_run и cross\_count в начальные значения. Это глобальные переменные, поэтому перед попыткой изменить их значение, в Python надо использовать ключевое слово global.

Текст функции:

def new\_game():

for row in range(3):

for col in range(3):

field[row][col]['text'] = ' ' field[row][col]['background'] = 'lavender'

global game\_run

game\_run = True

global cross\_count

cross\_count = 0

Функция click() будет вызываться после нажатия на поле, то есть при попытке поставить крестик. Функция выполняет следующие действия:

• Если игра еще не завершена, то крестик ставится. После этого увеличиваем счетчик количества выставленных крестиков cross\_count.

• Потом проверяем с помощью функции check\_win(), не победили ли мы этим ходом. Если еще не выявлен победитель и есть еще ходы, то выполняет ход компьютер функцией computer\_move(), и также после хода идет проверка выигрыша.

def click(row, col):

print(row,' ',col)

if game\_run and field[row][col]['text'] == ' ':

field[row][col]['text'] = 'X'

global cross\_count

cross\_count += 1 check\_win('X')

if game\_run and cross\_count < 5:

computer\_move()

check\_win('O')

Проверка победы Функция check\_win() осуществляет проверку выигрыша. Функция выполняет следующие действия:

• Она перебирает все возможные комбинации полей, образующих линию и вызывает с ними функцию check\_line(). Переменная smb – это символ «X» или «O», то есть крестики или нолики. Если задан «O», то проверяется не победил ли компьютер, а если задан «X», то проверяется не победил ли игрок.

• Если зафиксирован выигрыш, то меняем цвет фона кнопок, составляющих линию на розовый (pink). А также записываем в game\_run значение False.

Текст функций:

def check\_win(smb): for n in range(3):

check\_line(field[n][0], field[n][1], field[n][2], smb)

check\_line(field[0][n], field[1][n], field[2][n], smb)

check\_line(field[0][0], field[1][1], field[2][2], smb)

check\_line(field[2][0], field[1][1], field[0][2], smb)

def check\_line(a1,a2,a3,smb):

if a1['text'] == smb and a2['text'] == smb and a3['text'] == smb:

a1['background'] = a2['background'] = a3['background'] = 'pink'

global game\_run

game\_run = False

Действия компьютера Ход компьютера рассчитывается в функции computer\_move(). Алгоритм его действий следующий:

1. Проверка возможности победы компьютера. Если компьютеру представился шанс победы – он не должен его упустить. Сразу же делает победу.

2. Проверка возможной победы противника за один ход. Если игрок выставил два крестика в ряд, компьютер пытается разрушить планы игрока.

3. Случайный ход. Так как победить нет возможности и нет угрозы проигрыша, то выбирается случайное свободное поле. В бесконечном цикле while перебираются случайные числа, пока они не выпадут на не занятое поле. Для реализации этого алгоритма используется вспомогательная функция can\_win(), которая определяет возможность победы одного из игроков, возвращая значение False или True в переменной res.

Текст функций:

def can\_win(a1,a2,a3,smb):

res = False if a1['text'] == smb and a2['text'] == smb and a3['text'] == ' ':

a3['text'] = 'O' res = True if a1['text'] == smb and a2['text'] == ' ' and a3['text'] == smb:

a2['text'] = 'O' res = True if a1['text'] == ' ' and a2['text'] == smb and a3['text'] == smb:

a1['text'] = 'O' res = True return res

def computer\_move(): # Проверка возможности победы компьютера

for n in range(3):

if can\_win(field[n][0], field[n][1], field[n][2], 'O'):

return

if can\_win(field[0][n], field[1][n], field[2][n], 'O'):

return

if can\_win(field[0][0], field[1][1], field[2][2], 'O'):

return

if can\_win(field[2][0], field[1][1], field[0][2], 'O'):

return # Проверка возможности победы компьютера

for n in range(3):

if can\_win(field[n][0], field[n][1], field[n][2], 'X'):

return

if can\_win(field[0][n], field[1][n], field[2][n], 'X'):

return

if can\_win(field[0][0], field[1][1], field[2][2], 'X'):

return

if can\_win(field[2][0], field[1][1], field[0][2], 'X'):

return # Случайный ход

while True:

row = random.randint(0, 2)

col = random.randint(0, 2)

if field[row][col]['text'] == ' ':

field[row][col]['text'] = 'O'

break

**Графический интерфейс**

Все элементы графического интерфейса размещаются с помощью упаковщика grid. В цикле добавляются кнопки игрового поля. Они будут храниться в двумерном списоке field. В языке программирования Python добавляют элементы в список с помощью метода append.

Свойство colorspan у кнопки «Начало игры» выставляем в 3, чтобы он занимал всю ширину игрового поля.

for row in range(3):

line = [] for col in range(3):

button = Button(root, text=' ', width=4, height=2,

font=('Verdana', 20, 'bold'),

background='lavender',

command=lambda row=row, col=col: click(row,col))

button.grid(row=row, column=col, sticky='nsew')

line.append(button)

field.append(line)

new\_button = Button(root, text='new game', command=new\_game)

new\_button.grid(row=3, column=0, columnspan=3, sticky='nsew')

root.mainloop()

Для того, чтобы начать игру, игрок должен нажать кнопку мышки на поле. При нажатии начинается игра, то есть вызывается метод game();.Первым всегда ходит игрок и фишкой – крестик.

АНАЛИЗ РЕЗУЛЬТАТОВ РАЗРАБОТКИ ПРОГРАММЫ

Для тестирования работы программы были использованы различные варианты расположения фишек на различных полях. Программа работает правильно без  явных сбоев. В таблице рекордов программа корректно выполняет считывание данных  из файла и производит запись или перезапись  данных в файл.

ЗАКЛЮЧЕНИЕ

В ходе написания программы были использованы элементы объектно-ориентированного программирования. Основу программы составили классы. Было использовано переопределение стандартных операций для переменных абстрактных типов, а так же работа с графикой, строками и т.д.

Были применены знания, полученные в процессе изучения курса «Программирование на ЯВУ». При выполнении работы была проработана литература по языку Python, объектно-ориентированному программированию. Работать с объектами класса при помощи функций членов намного удобнее, нежели не использовать классы, а применять только стандартный набор типов данных.

Пользователю не придется тратить много времени на освоение правил игры «Крестики-нолики». Простота и наглядность интерфейса делают игру удобной в использовании.

Выполнение курсовой работы помогло закрепить материал и навыки, полученный на лекциях и лабораторных работах, помогло систематизировать знание языка и научится применять их к конкретным задачам.