1. Introdução

Desde que se começou os estudos sobre linguagens de programação lá nos anos 50, foram desenvolvidos vários paradigmas de programação, que podem ser definidos como a forma como a solução de um dado problema lógico é abordado. Diferentes paradigmas podem resolver o mesmo problema, mas cada uma delas tem as suas vantagens e desvantagens, e ter conhecimento sobre elas é vantajoso para que se possam desenvolver soluções com o máximo de eficiência possível.

Há um extenso leque de opções, independente da escolha da linguagem, todas são direcionadas a um objetivo no qual o programador deseja, tendo como base de estudos a programação é um processo de escrita e uma padronização de funções pré-definidas para o uso da área de tecnologia. Por conta do avanço da tecnologia temos hoje em uso a programação em diversas áreas distintas como a área da medicina, computação e principalmente no dia a dia, estando presente atualmente até em eletrodomésticos.

Dado o tema do trabalho, foi sugerido que houvesse o desenvolvimento de um mesmo jogo utilizando dos diferentes paradigmas imperativos apresentados em aula, para que houvesse um mais profundo entendimento do porquê há tantas linguagens distintas no mercado e para que se pudesse também mostrar as diferenças que a escolha de um paradigma pode surtir no projeto.

2. Objetivos

2.1. Objetivo Geral

Conseguir maior entendimento sobre os principais paradigmas de linguagens de programação apresentadas em sala e desenvolver 3 aplicações.

2.2. Objetivos Específicos

- Entender e descrever as linguagens imperativas e suas derivadas.
- Entender e descrever as linguagens descritivas e suas derivadas.
- Desenvolver um jogo em C utilizando do paradigma estruturado.
- Desenvolver um jogo em Java utilizando do paradigma orientado a objetos.
- Desenvolver um jogo em Python utilizando de quaisquer paradigmas.

3. Modelos e Paradigmas de programação

Dentro dessa ciência, existem 2 principais modelos, e de cada um deles tem 2 paradigmas: o modelo imperativo, que engloba o paradigma procedural e o paradigma de orientado a objetos; e o modelo declarativo, que engloba o paradigma funcional e o paradigma lógico.

3.1. Modelo imperativo

Como o nome indica, ela dá comandos, ditando COMO o algoritmo deve funcionar, e estruturas de decisão e repetição são comuns. Ela descreve as etapas e processos que o algoritmo deverá seguir para chegar a dado resultado. Tem como principais linguagens o C e Java. Nos primórdios da programação, utilizava-se a tag GOTO, que na época supria bem as necessidades dos desenvolvedores, mas a falta de linearidade do programa prejudicava muito a legibilidade dele, e com o passar do tempo, soluções foram desenvolvidas para esse problema, e a mais famosa delas é o uso de funções, que deu origem ao paradigma procedural

3.1.1. Paradigma Procedural

Considerado o primeiro paradigma criado o paradigma procedural costuma ser as postas de entrada para os estudos sobre lógica de programação, pois é mais próximo da linguagem de máquina, ela é bem simples, ela permite que o programa agrupe partes do código em procedures, ou função como são mais comumente chamadas, de forma que uma mesma parte do código pode ser executada várias vezes durante a execução, tornando o código mais limpo. Pode também ser chamada de paradigma estruturado.

3.1.1.1. Vantagens

- Eficiência
- Muito predominante no mercado
- Fácil aprendizagem

3.1.1.2. Desvantagens

- Dificil legibilidade
- Linguagem costumam ser de baixo nível

3.1.2. Paradigma Orientado a Objetos

Tendo o Java como grande exemplo de linguagem que utiliza desse paradigma, ele se baseia no uso de classes e objetos em seu programa, que visam alcançar um maior nível de abstração dentro do programa, partindo sempre do princípio de que as suas características e métodos se baseiam em objetos da vida real. É considera por muitos a evolução do paradigma procedural, pois no geral, ela acaba tendo todas as vantagens do paradigma procedural e um pouco mais, além de conseguir facilmente representar objetos da vida real.

3.1.2.1. Vantagens

- Nível de abstração muito elevado
- Facilidade na reutilização de código

3.1.2.2. Desvantagens

- Tempo de execução maior do que no paradigma procedural
- Exige mais experiencia por parte do programador

3.2. Modelo Declarativo

Pode ser definido como o modelo que tenta dizer o O QUE o programa quer fazer ao invés do como, e isso fica bem claro em linguagens como o SQL, onde o programa mais parece uma frase do que um algoritmo. Além do SQL, outro exemplo de linguagens que utilizam desse modelo são programas de marcação, como o HTML e XML

3.2.1. Paradigma Funcional

A primeira linguagem a utilizar desse paradigma foi o Lisp na década de 50, no entanto, com a adição de muitas características imperativas, ela já não pode ser considerada uma linguagem funcional pura. Uma linguagem que mais atual que que se encaixa melhor com esse título é o Haskell. A principal característica desse paradigma é que nela você utiliza de funções que muitas vezes já vem com o programa ou que vem em um pacote externo. Um exemplo simples é a função sort() em arrays em python, onde não é necessário a criação de uma lógica imperativa para se fazer ordenação dos itens dentro do array.

3.2.2. Paradigma Lógico

O paradigma lógico difere muito das outras, e a principal linguagem desse paradigma é o Prolog, que foi a grande responsável pelo desenvolvimento da inteligência artificial como conhecemos hoje. Ela se baseia no uso de fatos para conseguir responder perguntas que são inputadas pelo usuário.

4. Desenvolvimento

4.1. Ideia do jogo e Regras

A ideia e regras do jogo foram todas tiradas do jogo de corrida de carros do programa Roda a Roda Jequiti, televisionado pela SBT e apresentado por Silvio Santos e suas filhas. As Regras do jogo são as seguintes: existem 3 carros de cores distintas em uma pista, onde os 3 começam na casa de número 0, e vence o carro que chegar na casa número 6 primeiro. Os carros andam de acordo com os números escolhidos em uma tabela que contêm 18 números, e cada número tem uma bolinha que representa um dos 3 carros. O carro que corresponder a bolinha revelada anda 1 casa, e assim o jogo progride.

Imagem 1 – início da corrida

A lógica desse jogo se transcreveu para a linguagem de programação da seguinte forma: a pista de corrida é representada por 3 variáveis de número inteiro A, B e C, todas começando em 0. Já a tabela de 18 números é representada por uma lista contendo 18 valores, que são preenchidas aleatoriamente com um número igual de caracteres ‘A’, ‘B’ e ‘C’, que representa as cores das bolinhas. No caso são 6 bolinhas de cada cor.

Imagem 2 - fím da corrida

Por último, existe um painel utilizado para a visualização do usuário, que não pode ver as cores das bolinhas antes do número ser selecionado, então para isso é utilizado uma lista de 18 valores inteiros representando o número do item na tabela, que no caso é apenas uma contagem de 0 a 17.

Em resumo, o jogo funciona dentro de um while, que a cada iteração verifica se a variável A, B ou C chegou no número 6, caso sim, o jogo encerra e é mostrado o vencedor, caso contrário, o jogo pede para que o usuário selecione um dos números que ainda não foram selecionados.

4.2. Ferramentas utilizadas

- Clion 2022.2.3
- Pycharm Community Edition 2021.3
- IntelliJ IDEA 2022.2.3

4.3. Desenvolvimento do código

4.3.1. Python - Multiparadigma

Foi a primeira a ser desenvolvida, pelo fato do python ser mais facil de desenvolver e a maioria da equipe tem mais experiencia com essa linguagem. Por conta de ter sido a primeira, ela foi a que mais demorou, pois foram alteradas muitas coisas durante a programação. Adotou-se para a solução desse problema o uso de várias funções para organizar o código, sendo elas: criaPainel(), pickNumero(), além da main().

De forma simples, o criaPainel inicializa todos os arrays e variáveis necessárias para a execução do programa, enquanto o pickNumero é usado para fazer a verificação do array e posterior modificação dele de acordo com o número escolhido pelo usuário.

4.3.1.1. Código

```
import random


def criaPainel():
    painel = []
    painel_front = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
    aleatorio = 0
    a = 0
    b = 0
    c = 0
    for i in range(18):
        while True:
            aleatorio = random.randint(1, 3)
            if (aleatorio == 1 and a < 6):
                painel.append(aleatorio)
                a += 1
                break
            elif (aleatorio == 2 and b < 6):
                painel.append(aleatorio)
                b += 1
                break
            elif (aleatorio == 3 and c < 6):
                painel.append(aleatorio)
                c += 1
                break
    return painel, painel_front


def pickNumero(numero, a, b, c):
    if painel[numero] == 1:
```

```
            painel_front[numero] = 'A'
            a += 1
        elif painel[numero] == 2:
            painel_front[numero] = 'B'
            b+= 1
        elif painel[numero] == 3:
            painel_front[numero] = 'C'
            c += 1
        painel[numero] = 0
        return painel_front[numero], a, b, c

if __name__ == '__main__':
    a = 0
    b = 0
    c = 0
    check = 'N/A'
    painel, painel_front = criaPainel()

    while True:
        if a == 6 or b == 6 or c == 6:
            break
        print("ULTIMA LETRA TIRADA: " + str(check))
        print(painel_front)
        print("A: " + str(a))
        print("B: " + str(b))
        print("C: " + str(c))
        while True:
            numero = int(input("Digite o número: "))
            if painel[numero] == 0:
```

```
            print("Digite um número que não foi escolhida")
        elif numero < 0 or numero > 17:
            print("Digite um número que esteja no painel")
        else:
            break
    check,a,b,c = pickNumero(numero, a, b, c)
    print("_____")
    print("\n"*40)
if a == 6: result = 'A'
elif b == 6: result = 'B'
elif c == 6: result = 'C'
print("O vencedor da corrida foi: " + result)
```

4.3.2. C - Estruturado

O funcionamento dele é muito semelhante ao do programa em python, a diferença é que nela não se foi utilizado funções, pois o C não consegue retornar arrays em suas funções, o que tornou muito dificil a implementação de funções, portanto decidiu-se fazer o programa todo em apenas um bloco.

4.3.2.1. Código

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
   char painel[18];
   char painel_front[18][3] =
{"0","1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17"};
   int contA = 0, contB = 0, contC = 0, cont = 0, aleatorio = 0;
```

```
setbuf(stdout, 0);
srand(time(0));

while (contA != 6 || contB != 6 || contC != 6) {
   while(1)
   {
      aleatorio = rand() % 3;
      if (aleatorio == 0 && contA < 6) {
         painel[cont] = 'A';
         contA += 1;
         cont += 1;
         break;
      } else if (aleatorio == 1 && contB < 6) {
         painel[cont] = 'B';
         contB += 1;
         cont += 1;
         break;
      } else if (aleatorio == 2 && contC < 6) {
         painel[cont] = 'C';
         contC += 1;
         cont += 1;
         break;
      }
   }
}

int a = 0, b = 0, c = 0, numero = 0;
char check = ' ';
```

```
while(1){
    if(a == 6 || b == 6 || c == 6){
        break;
    }
    printf("ULTIMA LETRA TIRADA: %c\n",check);
    for(int i = 0; i < 18; i++){
        printf("%s ",painel_front[i]);
    }
    printf("\nA: %d",a);
    printf("\nB: %d",b);
    printf("\nC: %d",c);

    while(1) {
        printf("\nDigite um numero: ");
        scanf("%d", &numero);
        if (numero < 0 || numero > 17) {
            printf("\nDigite um numero que esteja no painel");
        } else if (painel[numero] == 'X') {
            printf("\nDigite um numero que nao foi escolhida");
        } else {
            break;
        }
    }
    if(painel[numero] == 'A'){
        strncpy(painel_front[numero],"A",2);
        check = 'A';
        a += 1;
    }else if(painel[numero] == 'B'){
        strncpy(painel_front[numero],"B",2);
```

```
            check = 'B';
            b += 1;
        }else if(painel[numero] == 'C') {
            strncpy(painel_front[numero],"C",2);
            check = 'C';
            c += 1;
        }
        painel[numero] = 'X';
        printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
    }
    if(a == 6){
        printf("O vencedor da corrida foi: A");
    }else if(b == 6){
        printf("O vencedor da corrida foi: B");
    }else if(c == 6){
        printf("O vencedor da corrida foi: C");
    }
    scanf("%x");
    return 0;
}
```

4.3.3. Java – Orientado a Objetos

O último a ser desenvolvido, também foi o mais rápido, pois além de já se ter bem estruturado quais que seriam os métodos aplicados no código, ainda houve a vantagem da sintaxe da linguagem C e Java ser bastante semelhante, então foi possível a reutilização de grande parte do código anterior. O programa conta com 2 classes além da main, sendo eles o Painel, PainelFront e Placar.

4.3.3.1. Classe Main

```
import java.util.Scanner;

public class Main {
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Placar placar = new Placar(0,0,0);
    Painel painel = new Painel();
    PainelFront painelfront = new PainelFront();
    char check = ' ';
    int numero = 0;

    painel.criaPainel();
    painelfront.criaPainel();

    while(true){
        if(placar.getA() == 6 || placar.getB() == 6 || placar.getC() == 6){
            break;
        }
        System.out.println("ULTIMA LETRA TIRADA: "+check);
        for(String i : painelfront.getPainel()) {
            System.out.print(i + " ");
        }
        System.out.println("\nA: "+placar.getA());
        System.out.println("B: "+placar.getB());
        System.out.println("C: "+placar.getC());

        while(true) {
            System.out.print("Digite um numero: ");
            numero = sc.nextInt();
            if (numero < 0 || numero > 17) {
                System.out.println("Digite um numero que esteja no painel");
            } else if (painel.getLetra(numero) == 'X') {
```

```
            System.out.println("Digite um numero que nao foi escolhida");
        } else {
            break;
        }
    }
    if(painel.getLetra(numero) == 'A'){
        painelfront.atualizaPainel(numero,"A");
        check = 'A';
        placar.atualizaPlacar('A');
    }else if(painel.getLetra(numero) == 'B'){
        painelfront.atualizaPainel(numero,"B");
        check = 'B';
        placar.atualizaPlacar('B');
    }else if(painel.getLetra(numero) == 'C') {
        painelfront.atualizaPainel(numero,"C");
        check = 'C';
        placar.atualizaPlacar('C');
    }
    painel.usaLetra(numero);
    System.out.println("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
  }
  if(placar.getA() == 6){
    System.out.println("O vencedor da corrida foi: A");
  }else if(placar.getB() == 6){
    System.out.println("O vencedor da corrida foi: B");
  }else if(placar.getC() == 6){
    System.out.println("O vencedor da corrida foi: C");
  }
}
```

```
}
```

4.3.3.2. Classe Painel

```
import java.util.Random;


public class Painel {
   private char[] painel = new char[18];

   public void criaPainel(){
      Random rand = new Random();
      int contA = 0, contB = 0, contC = 0, aleatorio = 0, cont = 0;
      while (contA != 6 || contB != 6 || contC != 6) {
         while(true)
         {
            aleatorio = rand.nextInt(3);
            if (aleatorio == 0 && contA < 6) {
               this.painel[cont] = 'A';
               contA += 1;
               cont += 1;
               break;
            } else if (aleatorio == 1 && contB < 6) {
               this.painel[cont] = 'B';
               contB += 1;
               cont += 1;
               break;
            } else if (aleatorio == 2 && contC < 6) {
               this.painel[cont] = 'C';
               contC += 1;
               cont += 1;
               break;
```

```
            }
          }
        }
    }

    public char getLetra(int indice){
        return this.painel[indice];
    }

    public void usaLetra(int indice){
        this.painel[indice] = 'X';
    }
}
```

4.3.3.3. Classe PainelFront

```
public class PainelFront {
    private String[] painelfront = new String[18];

    public void criaPainel(){
        for(int i = 0;i < 18;i++){
            painelfront[i] = String.valueOf(i);
        }
    }

    public void atualizaPainel(int indice, String letra){
        this.painelfront[indice] = letra;
    }

    public String[] getPainel(){
```

```
        return this.painelfront;
    }
}
```

4.3.3.4. Classe Placar

```
public class Placar {
    private int a;
    private int b;
    private int c;

    public Placar(int a, int b, int c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public void atualizaPlacar(char letra){
        if(letra == 'A'){
            this.a = this.a + 1;
        } else if (letra == 'B') {
            this.b = this.b + 1;
        } else if (letra == 'C') {
            this.c = this.c + 1;
        }
    }

    public int getA()
    {
        return a;
    }
```

```
    public int getB()
    {
        return b;
    }
    public int getC()
    {
        return c;
    }
}
```

5. Conclusão

Dados os fatos apresentados, utilizando de todas as boas práticas da programação e seguindo as normas definidas pela comunidade, conclui-se que todos os objetivos propostos foram realizados. Assim podendo ter o estudo em diferentes linguagens (Python, Java e C) porém com o mesmo objetivo de criação de um jogo (jogo de corrida) e tendo ainda a possibilidade de entender novos e diferentes mecanismo de programação, e ainda teve se estudo sobre a história da programação e entender todo o processo de avanço da programação em nossas vidas e em usos em diferentes áreas. Vemos que a utilização da programação esta interligada aos avanços de tecnologia e necessidades do dia a dia. Com tudo podemos observar que funções diferentes e necessidades é necessário, a escolha de uma linguagem especifica para facilitar seu uso, porém se dá também o uso de várias linguagens para a realização de uma única função (com programações diferentes).

Referencias

PACIEVITCH, Yuri. História da Programação. In: PACIEVITCH, Yuri. História da Programação. [S. l.], 20 out. 2022. Disponível em: https://www.infoescola.com/informatica/historia-da-programacao/. Acesso em: 20 out. 2022.

DIGITAL HOUSE. Paradigmas de programação: saiba quais são as mais usadas e como solucionar problemas. In: DIGITAL HOUSE. Paradigmas de programação: saiba quais são as mais usadas e como solucionar problemas. [S. l.], 12 abr. 2022. Disponível em: https://www.digitalhouse.com/br/blog/paradigmas-de-programacao/#:~:text=O%20que%20%C3%A9%20paradigma%20de,seguir%20algumas%20regras%20quando%20implementadas. Acesso em: 20 out. 2022.

COSTA, Sergio. Uma visão muito breve sobre o paradigma funcional. In: COSTA, Sergio. Uma visão muito breve sobre o paradigma funcional. [S. l.], 10 dez. 2015. Disponível em: https://sergiocosta.medium.com/paradigma-funcional-3194924a8d20. Acesso em: 24 out. 2022.

NOLETO, Cairo. Programação Funcional: o que é, principais conceitos e vantagens!. In: NOLETO, Cairo. Programação Funcional: o que é, principais conceitos e vantagens!. [S. l.], 16 ago. 2020. Disponível em: https://blog.betrybe.com/tecnologia/programacao-funcional/. Acesso em: 27 out. 2022.

VIEIRA, Leandro Fernandes. Programação Funcional: o que é, principais conceitos e vantagens!. In: VIEIRA, Leandro Fernandes. Programação Funcional: o que é, principais conceitos e vantagens!. [S. l.], 24 dez. 2015. Disponível em: https://leandromoh.gitbooks.io/tcc-paradigmas-de-programacao/content/index.html. Acesso em: 22 out. 2022.

SEBESTA, Robert W. CONCEITOS DE LINGUAGENS DE PROGRAMAÇÃO. 9. ed. [S. l.]: Bookman, 2011.