# GAME

Carolina Leite e Magali Cortes*
*Universidade Federal de Santa Catarina (UFSC)

***Resumo*—Apresentamos o processo criativo de um jogo em linguagem C de programação, fazendo uso do editor Kate disponível no sistema operacional Linux. Abordamos todos os blocos escritos necessários para compilar o programa.**

***Palavras-chave*—Jogo, linguagem de programação, linguagem C, matriz, vetor, estrutura de repetição, terminal..**

***Abstract*—We introduce the creation of a game in programming language C, making use of the Kate editor available on the Linux operating system. We cover all written blocks needed to compile the program.**

***Index Terms*—Game, programming language, C language, matrix, vector, repetition structure, terminal.**

## I. Introdução

ESTE projeto foi proposto pelo professor da disciplina, usado como método didático alternativo. O objetivo foi uma maior compreensão de lógica de programação, tipos de dados, estruturas de seleção e repetição, funções e estruturas de dados e outros tópicos abordados na disciplina de programação I. No desenvolvimento do jogo passamos pelo processo criativo, de produção e documentação. Tivemos apoio dos monitores e do professor ao longo do semestre. O código de um game pode apresentar blocos muito complexos, usando de diversos recursos, como efeito sonoro usado em nosso jogo.

## II. Fundamentação teórica

### A. Histórico dos games

Foi no período da guerra fria, com a necessidade de alguma distração prazerosa, que um cientista norte americano pensou em usar a mesma lógica usada como ferramenta de guerra, para criar um tipo de jogo interação homem-máquina. Apenas em 1972 foi disponibilizado um jogo com objetivo comercial, o Pong. Em seguida alguns jogos famosos como Pacman(1980) e Donk Kong (1981) foram surgindo. No Brasil, o primeiro jogo foi o Telejogo em 1977.

### B. Lógica de programação

Pode-se dizer que lógica - de maneira geral - é construir a solução ou definir a correção de algum problema ou conceito. E ela envolve algum tipo de análise. Pode ser simples, como por exemplo definir que morango é uma fruta e não um legume - ou, pode ser complexa como a resolução de algum cálculo avançado.

A lógica de programação se diferencia por conta da linguagem específica que usa (existem diversas) e pelo seu objetivo específico de resolver problemas digitais.

Universidade Federal de Santa Catarina (UFSC). Correspondência ao autor: Carolina Leite e Magali Cortes (email: seuemail@grad.ufsc.br).

Um programa é uma sequência lógica de comandos que buscam solucionar algum problema. O princípio da lógica desenvolvida no programa pode envolver um fluxograma, que é uma ferramenta gráfica para apresentação do algoritmo. Por sua vez, o algoritmo é uma sequência de passos resumidos que guiam a idéia principal para o programa. Na imagem a seguir está um fluxograma para solução do problema: trocar lâmpada.

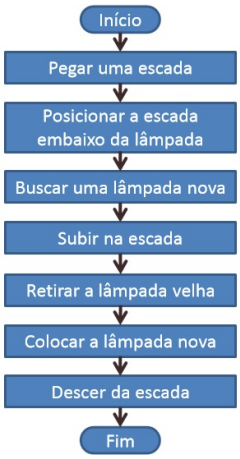


Figura 1. Exemplo de fluxograma.

O algoritmo é também importante para o mundo da programação, pois ele independe da linguagem escolhida, ou seja, pode ser interpretado de forma universal.

### C. Dados

Diferentes tipos de dados são a forma como recebemos e manipulamos informações em um programa. Existem quatro tipos primitivos, que são:
o tipo inteiro - conjunto dos números inteiros ...-2,-1,0,1,2...; tipo real - conjunto dos números reais ...-3/2, -0.5, . . .; caracter - caracteres alfanuméricos 9, a, ; lógico - é uma operação que pode gerar dois resultados diferentes E, OU, NÃO.

Para utilizar dados em um programa, precisamos declarar antes. Podemos guardar ou não, os dados, conforme a operação exigir.

A entrada de dados se dá por um comando de leitura, em nosso caso o " printf(..) " e a saída por um comando de escrita, o "scanf(...)".

### D. Estruturas de controle

Um programa é composto pela entrada, saída e manipulação de dados. Esta última se dá através de comandos, que são

estruturas de controle em um formato padrão. A seguir esmiuçamos brevemente cada uma das estruturas básicas na construção de um código de programação.

Estrutura de seleção simples (“if” ) - é um teste básico.

Verifica se uma determinada condição é verdadeira e, em caso positivo, executa um comando. O exemplo de um programa que calcula a média aritmética das notas de um aluno e dá o resultado da aprovação, é mostrado a seguir:

```
início
   entrada de dados;
   processamento;
   se (as notas atingirem a média necessária)
            então mostre (“aluno aprovado”);
fim
```

Seleção composta - segue o princípio da anterior.

Porém, com a possibilidade de gerar um segundo comando em caso do teste apresentar resultado negativo.

```
início
    entrada de dados;
    processamento;
    se (as notas atingirem a média necessária)
            então mostre (“aluno aprovado”);
    senão
            mostre ("aluno não aprovado");
fim
```

Seleção encadeada

Quando o código requer mais de uma condição para executar uma função, pode-se testar várias vezes num mesmo bloco. Esta estrutura pode ser homogênea (quando apresentar um padrão lógico) ou heterogênea (quando não). A seguir um modelo de seleção encadeada heterogênea.

```
se
    então
        se
            então
                comandos;
            fim se
fim.
```

Estrutura de repetição (“while”, “do”, “for”)

A maioria dos códigos não são pensados para processar apenas um dado uma única vez. Em geral, são utilizadas repetições que permitem ao programador trabalhar com inúmeros dados e testes, de acordo com a capacidade de processamento e memória disponível. Como ilustrado no exemplo padrão a seguir:

```
início
   entrada de dados;
   enquanto (condição)
        bloco de comandos;
   fimenquanto;
fim
```

Além deste formato, podemos executar o teste no final:

```
início
   entrada de dados;
   faça
        bloco de comandos;
   até (condição);
fim
```

Algumas situações não requerem teste de condição, pode-se desejar apenas a repetição dos comandos um determinado número de vezes. Como exemplificado:

```
para x de xo até xf passo p faça
   comandos:
fimpara;
```

### *E. Estruturas de dados: Vetores e matrizes*

As estruturas de dados são um recurso que nos permitem construir um novo tipo de dado conforme a demanda de nosso projeto.

Por exemplo, suponha que fosse requisitado um código que analise as notas escolares de 500 alunos e depois retorne a quantidade de aprovados. Isso demandaria trabalhar com uma quantidade específica de dados.

O intrumento que poderíamos utilizar para resolver este problema, é o Vetor. Este, nos permite armazenar a quantidade exata de dados necessária. Previamente defimos o tamanho que ele possui, já na declaração. Como por exemplo na figura a seguir mostramos um vetor de tamanho 10, ou seja, ele pode armazenar dez valores.

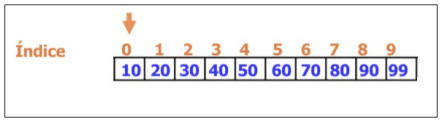


Figura 2. Vetor com 10 posições (vet[10]

Este é um vetor com 10 espaços ocupados por diferentes valores. Por exemplo, a posição 1, está preenchida pelo valor 20. Ou seja, acessando a posição um - vet[1] - estamos lendo seu conteúdo: 20.

Matrizes: Estrutura composta multidimensional

Uma construção de dados ainda mais sofisticada, é a matriz. Que nos permite transitar não apenas por espaços enfileirados, mas apresenta uma segunda dimensão, que são as colunas.

A figura a seguir representa uma matriz M[2][3].

Para acessar um espaço nesta estrutura de dados, precisaremos de dois laços de repetição: Um para percorrer

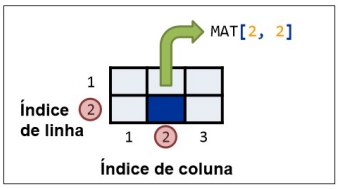


Figura 3. Matriz de 2 linhas e 3 colunas.

as linhas, e outro para percorrer as colunas.

Strings

Está relacionado a criação de imagens para serem visualizadas num determinado terminal utilizando caracteres. A string desejada pode, por exemplo, ser uma frase. Cria-se um vetor, onde cada caracter ocupa um espaço. Como representado na imagem.

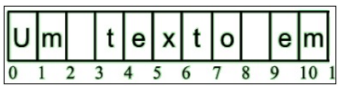


Figura 4. Exemplo visual de uma string.

A biblioteca <string.h> possuí diversas funções que nos permitem trabalhar com estas sentenças. Tais quais:

- copiar uma string para a outra;
- obter o tamanho da string;
- comparar duas strings;
- ler uma string do teclado.

## III. Metodologia

A partir de uma proposta bem definida pelo professor da disciplina, utilizamos os seguintes métodos e ferramentas para o desenvolvimento do projeto.

### A. Editor KATE

A ferramenta KATE é um editor avançado multi documento criado em 2000. Permite ao usuário programar, compilar e executar. Algumas das vantagens de sua utilização são:

- a possibilidade de usar diversas linguagens de programação;
- o acesso ao editor e o terminal (onde aparece a execução do programa);
- acessar diferentes arquivos ao mesmo tempo.

A figura 5 mostra a aparência do editor utilizado para criar o código de nosso *game*. A parte branca foi utilizada para escrita, e os resultados, após a compilação, são vistos na parte mais escura.

### B. LINUX

Em nosso projeto, trabalhamos exclusivamente com o sistema operacional LINUX, pois foi uma exigência.

O sistema foi lançado no ano de 1991 - passando sempre por atualizações até hoje - pelo engenheiro de software finlandês, Linus Torvalds.

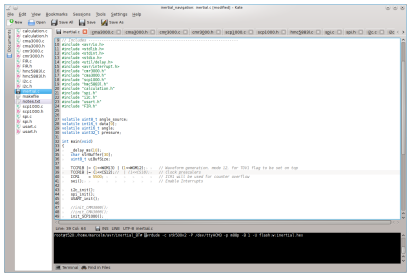

Figura 5. Página inicial e terminal do jogo.

Em essência, Linux é o nome dado ao **Kernel** que é um componente que define instruções relacionadas ao processador, à memória e aos dispositivos periféricos - e junto de outros programas essênciais constitui um sistema operacional.

Uma característica do Linux são as distribuições, que são variações do sistema. Embora ele permaneça com o mesmo fundamento, sendo o mesmo sistema operacional, pode ter aparencias e formas voltadas para determinados nichos. As diferenças que caracterizam as distribuições, se dão por softwares diversos como editores de texto, navegadores e interfaces gráficas.

### C. Desenvolvimento do game

Nosso jogo foi baseado no jogo do navegador google, que é disponibilizado quando não há conexão com a internet. Este é um jogo onde o personagem que controlamos precisa atravessar obstáculos, desviando-se por cima ou por baixo.

Chamamos nosso jogo de CTJump. E a proposta dele foi fazer algo parecido com acima citado, porém mais desafiador. Nosso jogo contou com mais personagens que precisam passar pelos mesmo obstáculos, quase simultâneamente.A princípio, os três personagens que criamos, são controlados por apenas um jogador.

## IV. Resultados

Foram obtidos os resultados esperados. Conseguir desenvolver o jogo conforme requisitado e testá-lo várias vezes, com sucesso. Utilizando dos conceitos passados em sala de aula, e com a ajuda dos monitores da disciplina, desenvolvemos o código no programa editor de texto *kate*.

A seguir uma imagem da tela inicial, mostrando o nome e o menu de opções do jogo.


Figura 6. Página inicial e terminal do jogo.

Ao desenvolver, encontramos dificuldades diversas, principalmente pela pouca expêriencia com lógica e por ser um

projeto muito desafiador que demanda técnicas complicadas para nosso nível iniciante, por exemplo, na ampla manipulação de matrizes que o terminal do jogo demandou.

A seguir uma imagem do nosso *game* pausado.

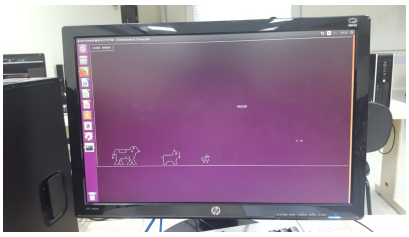

Figura 7. Página inicial e terminal do jogo.

Os resultados foram muito satisfatórios.

Nesta seção são apresentados os resultados de simulação e/ou experimentas do seu trabalho. Caso não tenham realizados experimentos ou obtidos resultados concretos, pode-se descrever os resultados esperados.

## V. Conclusão

A lógica desenvolvida com os fundamentos teóricos trouxe-nos um novo nível de capacitação. Além do instigamento da criatividade que pode representar um perfil empreendedor para o aluno da disciplina.

## Referências

[1] Forbellone, Andre Luiz Villar *Lógica de programação - A construção de algoritmos e estrutura de dados*, 3rd ed. Pearson prentice hall São Paulo: 2005.