JOGOS DIGITAIS: Aplicação da Estrutura de Dados com Lista Circular

Hélder Alves Couto
Discente do Curso Engenharia da Computação
Centro Universitário de Votuporanga (UNIFEV)

Renato Gomes Marcacini
Discente do Curso Engenharia da Computação
Centro Universitário de Votuporanga (UNIFEV)

Fernando Menechelli
Docente do Curso Engenharia da Computação
Centro Universitário de Votuporanga (UNIFEV)
Especialista em Banco de Dados

RESUMO

Este projeto tem como objetivo realizar um levantamento bibliográfico referente às características e o contexto da linguagem de programação no uso de Listas Simplesmente Encadeadas Dinâmicas. É abordado o conceito de programação da linguagem C, a estrutura de programação no uso de Listas Circulares e um exemplo de código da Linguagem C no uso de Lista Encadeada. O artigo discute ainda o contexto da alocação de memória, limite, ponteiro, uma síntese no uso de Listas Dinâmicas e aplicações que utilizam este recurso, tais como jogos eletrônicos.

PALAVRAS-CHAVE: Lista Circular, Estrutura de Dados, Jogos.

INTRODUÇÃO

A linguagem de programação C é uma evolução de duas linguagens anteriores, BCPL e B. BCPL foi criada para a escrita de sistemas operacionais e compiladores. Destaca no quesito de sistemas operativos as versões iniciais do *kernel*[1] do *Unix*[2] que foram desenvolvidas inicialmente em linguagem B. Como B e BCPL não tinham em sua estrutura de programação a definição dos tipos de dados ficara a cargo do programador realizar esta tratativa de seus tipos, conforme DEITEL & DEITEL (2001, p.56).

Em linha de sucessão, a linguagem C foi idealizada por Dennis Ritchie e originalmente implementada em um computador em 1972. A linguagem traz consigo a introdução dos tipos de dados, ausente nas linguagens B e BCPL. Ficou conhecida como a linguagem de programação usada para implementação das novas

versões do *Kernel*[1] Unix[2] e de diversos núcleos de sistemas operativos da época, DEITEL & DEITEL (2001, p.56).

A linguagem aborda conceitos de programação importantes como Estruturas de Dados *Struct*[3], Pilha[4], Fila[5] e Lista[6], alocações estáticas de memória, ou seja, reserva de memória pré-fixada, alocações dinâmicas de memória, onde o limite da memória para alocação pelas aplicações é a própria memória existente e disponível no computador, LAUREANO (2008, p.83).

A motivação da pesquisa sobre a linguagem C no uso de Listas Circulares Dinâmicas foi devido à grande importância desprendida pelo tema, como seu limite de usabilidade, os métodos de alocação utilizados, estes fixos ou dinâmicos, o uso de ponteiros, sendo este uma variável que guardará um endereço de memória, ou seja, o ponteiro aponta para uma posição de memória no computador e a implementação da linguagem no desenvolvimento de aplicações como jogos digitais, conforme LAUREANO (2008, p.11).

O objetivo que levou a produção deste artigo foi elucidar tópicos relevantes no uso de Lista Circulares Dinâmicas. Desta forma a metodologia de pesquisa do artigo é um levantamento bibliográfico e está dividida em cinco seções. A primeira seção é destinada a definição da linguagem no uso de alocação estática versus alocação dinâmica e lista circular simplesmente encadeada. A segunda seção será abordada a aplicabilidade de uma Lista Circular.

As funcionalidades da Lista Circular ficam destinadas a terceira seção. A quarta seção descreve um caso de abstração de dados o chamado *struct* e por último a quinta seção aborda um trecho de código com especificações de como implementar esta estrutura em linguagem de programação C, aborda ainda a iniciação de uma lista, os métodos de inserção, como exibir, imprimir na tela, os dados de uma lista circular e a execução de uma aplicação com uso desta estrutura

---

[1]Kernel: é o núcleo do sistema operacional, responsável pela gerência dos recursos do hardware usados pelas aplicações, MAZIERO (2013, p.08).

[2]Unix: Sistema Operacional criado em 1969 por Ken Thompson e Dennis Ritchie, pesquisadores dos Bell Labs, MAZIERO (2013, P.25).

[3]Struct ou Estrutura de Dados: é uma estrutura ou registro, dependendo da linguagem de programação utilizada, designado por um grupo de itens, onde cada item tem um identificador próprio, sendo cada identificador, um membro, ou um campo da estrutura, LANGSAM, AUSGENSTEIN & TANENBAUM (1995, p.57).

[4]Pilha: é o inverso de uma fila porque usa o acesso último a entrar, primeiro a sair, o que as vezes é chamado de LIFO (Last In, First Out), SCHILDT (1996, P.535).

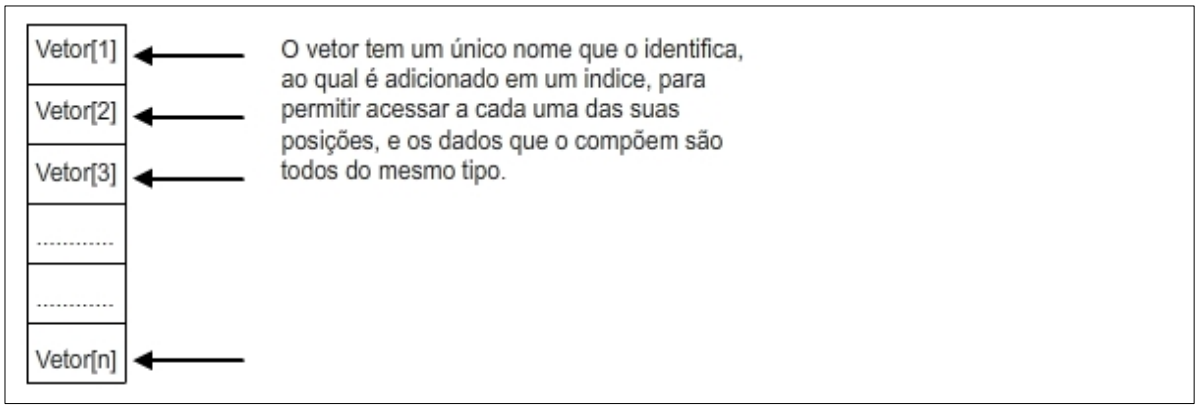
[5]Fila: é uma lista linear de informações, que é acessada na ordem primeiro a entra, primeiro a sair, SCHILDT (1996, P.526).

[6]Lista: é uma corrente que pode trabalhar com seu acesso de forma randômica, porque cada porção de informação carrega consigo um elo, ponteiro, ao próximo item de dados nesta corrente, SCHILDT (1996, P.540).

1. DEFINIÇÃO

De acordo com LAUREANO (2008, p.19), em uma alocação estática ao exemplo à declaração de um vetor, torna-se necessário dimensioná-lo, ou seja, prever a quantidade de espaço a ser ocupada pelo software durante a codificação. Tal definição é um fator limitante na dinâmica de espaços de memória do computador.

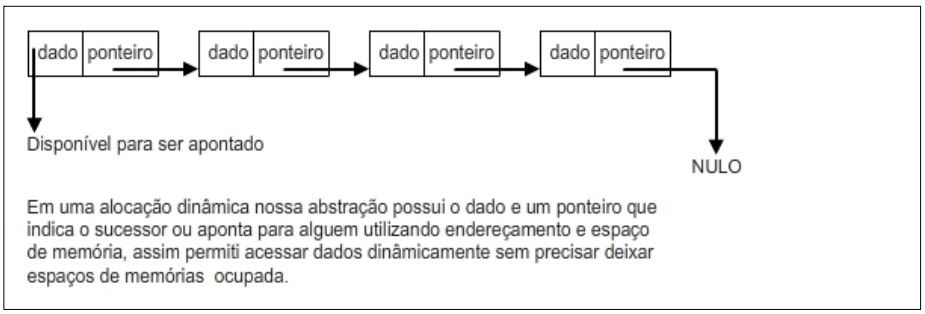
Figura 1: Exemplo ilustrado de uma alocação estática utilizando vetor.



Fonte: Adaptado de LAUREANO (2008, p.19).

A alocação dinâmica se assemelha a um vetor, porém sem desperdício de memória, isso se deve devido a requisição de memória ao sistema, em tempo de execução, um espaço de um determinado tamanho. Os dados se mantém reservados até serem liberados pelo programa, disponibilizando novos espaços mantendo o tamanho devidamente ocupado não fixo, LAUREANO (2008, P19).

Figura 2: Exemplo ilustrado de uma alocação dinâmica utilizando lista simples.



Fonte: Adaptado de LAUREANO, (2008, p.19).

> "Uma lista é uma estrutura de dados dinâmica. O número de nós de uma lista pode variar consideravelmente à medida que são inseridos e removidos elementos. A natureza dinâmica de uma lista pode ser comparada à natureza estática de um vetor, cujo tamanho permanece constante", TANEMBAUM (1995, p.225).

Desta forma uma estrutura é uma lista, onde cada elemento aponta para o seu sucessor e o último elemento aponta para o primeiro da "Lista". Cada elemento é tratado como um ponteiro que é alocado dinamicamente a medida das inserções ou remoções dos dados, para isso utiliza-se "ponteiro para ponteiro", fica assim mais fácil o manuseio para mudança do dado inicial, TANEMBAUM (1995, p.280). A Estrutura é flexível e dinâmica.

> "Listas são estruturas muito flexíveis porque podem crescer ou diminuir de tamanho durante a execução de um programa, de acordo com a demanda. Itens podem ser acessados, inseridos ou retirados de uma lista. Listas são adequadas para aplicações onde não é possível prever a demanda por memória, permitindo a manipulação de quantidades imprevisíveis de dados, de formato também imprevisível", ZIVIANI (1999, p.35).

O uso de Estrutura é vantajoso para economia de memória, vista que diferentemente de vetores estáticos, em aplicações o uso da lista se torna conveniente por não precisar de um tamanho ou previsão do crescimento da lista,

além de em tempo de execução é alojado ou liberado espaço de memória para novos dados, ZIVIANI (1999, p.38).

2. APLICAÇÃO EM JOGOS DIGITAIS

Conforme SAVI (2011, p.25), o desenvolvimento de jogos digitais é uma área complexa que exige muito estudo em diversas áreas tais como: programação, design, sonoplastia, arquitetura. Desta forma o uso da estrutura de dados deve andar lado a lado com a parte de produção tecnológica.

Devido à natureza cíclica de uma lista circular dinâmica, esta é uma das estruturas mais utilizadas na área de jogos, seu uso facilita a minimização de problemas como repetições intervaladas em jogos de natureza circular e regradas. Exemplo: jogos de cassinos.

Figura 3: Exemplo ilustrado jogo de cassino, Jackpot.

Fonte: https://www.yiv.com/Jackpot-777

O Jackpot é um jogo de cassino no qual é acumulado uma grande quantia em dinheiro dentro das máquinas, a forma de vencer o jogo é puxar a alavanca de tal forma que gira três bobinas, neste exemplo da Figura 3, todas devem retornar a célula 7 para o jogador ganhar o maior prêmio. Uma das soluções para realizar ou

construir esse maquinário seja de forma virtual ou mecânica está na utilização da lista circular, percebe-se que as bobinas giram de forma repetida e a cada novo jogo continuam em repetições intervaladas retornando um tipo de valor.

Figura 4: Exemplo ilustrado, jogo de cassino, Roulette (Roleta).

Fonte: http://www.clickjogos.com.br/jogos/roulette-royale/

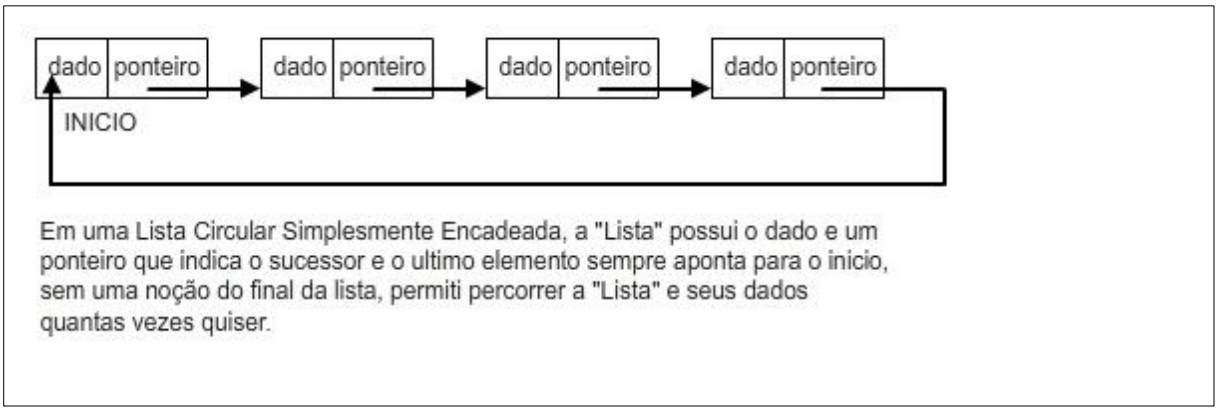
A Roullete (Roleta) como apresentado na Figura 4, é um jogo de cassino com o objetivo de apostar dinheiro em certa quantidade de números disponíveis. Após a aposta gira-se a roleta junto a uma bolinha em seu término do seu ciclo obtém o retorno de um valor, se o número apostado for de igual valor do retorno da roleta, o jogador vence, caso contrário a banca vence. Da mesma forma do jogo Jackpot apresentado acima, a roleta seja virtual ou mecânica também é solucionado devido a sua natureza cíclica e repetida, criando uma analogia: Neste caso a bolinha é o ponteiro da lista, que percorre os valores previamente inseridos na roleta. O impulso dado para girar a roleta é o valor até onde ela pode ser percorrida, que ficará em um laço de repetição até o retorno de seu valor.

3. FUNCIONALIDADE

Na implementação de uma lista circular dinâmica, sua composição é dada pelo tipo de estrutura ou dado que ela deve guardar, além de campos ou funções que controlam a própria lista conforme a aplicação necessita. Uma lista pode ter uma ou várias formas, o que torna sua funcionalidade bem ampla para diversas aplicações, LAUREANO (2008, p.79).

A utilização dos ponteiros em uma lista introduz a possibilidade de montar conjuntos de blocos de construção em estruturas flexíveis, podendo ser ligados, desligados e remontados durante a execução do programa, ou seja, quando for preciso, um bloco de espaço de memória ficará reservado para o ponteiro, e quando não mais necessário será liberado, assim agregando uma otimização, TANEMBAUM (1995, p.250).

Figura 5: Exemplo ilustrado de uma lista circular simplesmente encadeada.



Fonte: Adaptado de TANEMBAUM (1995, p.280).

Uma lista circular dinâmica facilita execuções em ciclos e que utilizam grandes quantidades de dados, isso se deve a natureza de seu comportamento cíclico e econômica na alocação de memória, comumente utilizada em aplicações com repetições intervaladas tais como apresentadas neste artigo.

4. ABSTRAÇÃO

Conforme LAUREANO (2008, p.16), uma *struct* é uma estrutura que contém diversas outras variáveis e normalmente de tipos diferentes, as variáveis internas

contidas são denominadas membros da *struct*. Podemos dizer que as *structs* da linguagem C são o equivalente ao que se denomina registros em outras linguagens de programação. Essas estruturas, podem ter quantos elementos você queira e dos tipos que você quiser definindo a estrutura e seus atributos, esses detalhes são definidos somente uma vez, na declaração da abstração.

O conceito de registro é muito utilizado quando temos elementos em nossos programas que precisam e fazem uso de vários tipos de variáveis e características. Utilizando, podemos trabalhar com vários tipos de informações de uma maneira mais fácil, rápida e organizada, uma vez que não temos que nos preocupar em declarar e decorar o nome de cada elemento contido, LAUREANO (2008, p.16).

Figura 6: Exemplo ilustrado de uma estrutura *struct*.

Fonte: LAUREANO (2008, p.17)

Figura 7: Exemplo de estrutura struct.

```
struct Funcionario
{
    char nome [40];
    struct
    {
        int dia;
        int mes;
        int ano;
    } dataNasc;
    char departamento[10];
    float salario;
};
```

Fonte: LAUREANO (2008, p.17).

5. DOCUMENTAÇÃO

Em uma lista circular existem quatro execuções básicas para sua funcionalidade, ela se ramifica em funções de inicialização, inserção, remoção e apresentação.

Para utilizar uma lista, primeiramente deve-se criar uma abstração da mesma, também conhecido como registro ou *Struct*. Dentro do *struct* deve-se conter o tipo de dado à inserção da lista e um ponteiro para auxiliar o apontamento para a próxima célula (item).

Figura 8: Exemplo de código, registro de uma lista circular.

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

struct no
{
    int dado;
    struct no *prox;
};

struct no *primeiro;
struct no *ultimo;
```

Fonte: Adaptado de LAUREANO (2008, p.88).

Na utilização de uma lista circular a lista deve ser inicializada com um apontamento *NULL* (nulo), pois ainda não há células/dados adicionados para auxílio.

Figura 9: Exemplo de código, inicialização de uma lista circular.

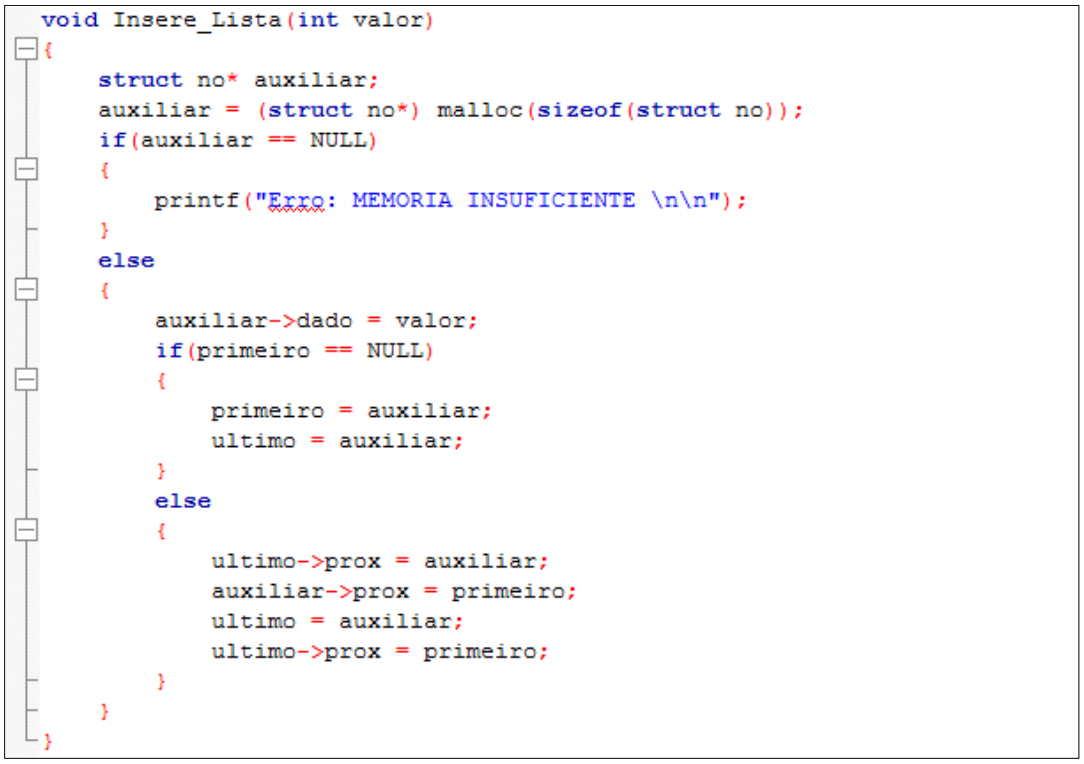
```
void Cria_Lista()
{
    primeiro = NULL;
    ultimo = NULL;
    printf("Lista Inicializada \n\n");
}
```

Fonte: Adaptado de LAUREANO (2008, p.88).

Em função de inserção à lista circular, cria-se uma nova lista de auxílio, ela é de extrema importância pois assim não se é manipulado dados da lista original, a

partir da auxiliar é feito a troca de apontamento, onde sempre a última célula aponta para primeira.

Figura 10: Exemplo de código, inserção de uma lista circular.

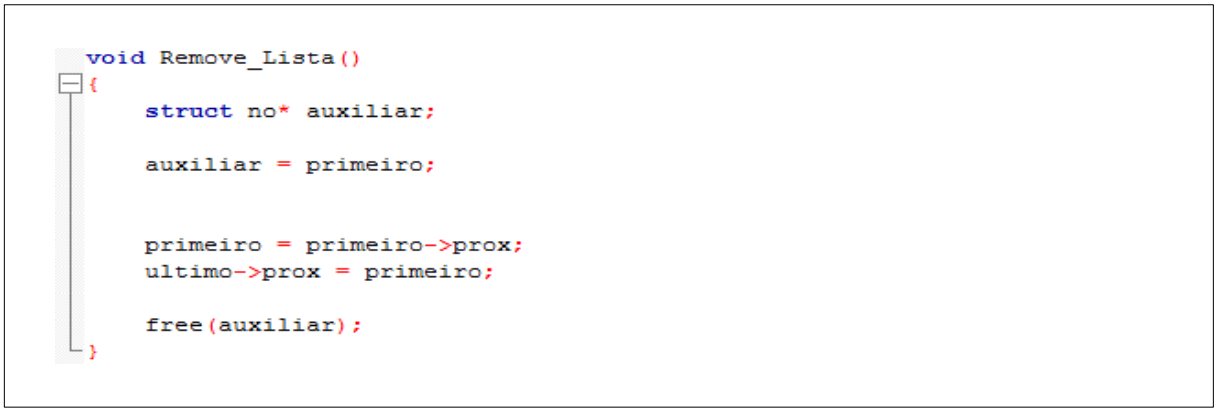


```
void Insere_Lista(int valor)
{
    struct no* auxiliar;
    auxiliar = (struct no*) malloc(sizeof(struct no));
    if(auxiliar == NULL)
    {
        printf("Erro: MEMORIA INSUFICIENTE \n\n");
    }
    else
    {
        auxiliar->dado = valor;
        if(primeiro == NULL)
        {
            primeiro = auxiliar;
            ultimo = auxiliar;
        }
        else
        {
            ultimo->prox = auxiliar;
            auxiliar->prox = primeiro;
            ultimo = auxiliar;
            ultimo->prox = primeiro;
        }
    }
}
```

Fonte: Adaptado de LAUREANO (2008, p.89).

Em função de remoção à lista circular, cria-se duas novas listas auxiliares para organização, no caso abaixo removendo-se a primeira célula da lista original, as auxiliares organizam para que a última célula aponte para uma nova célula inicial, mantendo-o circular.

Figura 11: Exemplo de código, remoção pelo início de uma lista circular.



```
void Remove_Lista()
{
    struct no* auxiliar;

    auxiliar = primeiro;


    primeiro = primeiro->prox;
    ultimo->prox = primeiro;

    free(auxiliar);
}
```

Fonte: Adaptado de LAUREANO (2008, p.90).

Em função de exibição à lista circular, cria-se uma nova lista auxiliar que possui exatamente os mesmos dados da lista original, assim evitando erros consequente à original. Utiliza-se um laço de repetição e uma quantidade de vezes que deseja percorrer a lista.

Figura 12: Exemplo de código, remoção pelo início de uma lista circular.

```
void Exibe_Lista(int termino)
{
    struct no *auxiliar;
    auxiliar = primeiro;

    int posicao = 0;
    while(posicao < termino)
    {
        printf("%i-> ", auxiliar->dado);
        auxiliar = auxiliar->prox;
        posicao++;
    }
    printf("\n\n");
    printf("Inicio: %i \n", primeiro->dado);
    printf("Ultimo valor apontando para: %i \n", ultimo->prox->dado);
}
```

Fonte: Adaptado de LAUREANO. (2008, p.91).

Após a execução das funções, o programa retornará uma lista circular dinâmica e ao final ocorre a liberação da memória com a função free() pertencente a linguagem C, mantendo-o dinâmico.

Figura 13: Execução do código adaptado.

```
->LISTA CRIADA

->VALOR INSERIDO NA LISTA: 1
->VALOR INSERIDO NA LISTA: 5
->VALOR INSERIDO NA LISTA: 2
->VALOR INSERIDO NA LISTA: 8
->VALOR INSERIDO NA LISTA: 3
APONTAMENTO:   INICIO 1->5->2->8->3->  INICIO 1->5->

->INICIO: 1
->FINAL: 3
->APONTAMENTO DA ULTIMA CELULA: 1
```

Fonte: Execução em Codeblocks pelos Autores.

CONSIDERAÇÕES FINAIS

Em razão do contexto abordado sobre Listas Circulares, foi possível elucidar quais são as caraterísticas relevantes no uso desta estrutura aplicado no desenvolvimento de jogos digitais e a dinâmica de alocação de espaço de memória e acesso a estes dados na memória do computador, desta forma é possível utilizar estas estruturas dinâmicas ao invés de estruturas estáticas devido a sua natureza de performance.

A lista circular dinâmica, facilita execuções em ciclos e que utilizam grandes quantidades de dados, isso ocorre pelo seu comportamento cíclico e ainda a sua economia na alocação de memória, caracterizando como uma alternativa para utilizações em aplicações com repetições intervaladas, conforme apresentadas na seção de jogos digitais, que utilizam ideais circulares e regrados.

REFERENCIAL BIBLIÓGRAFICO

CODEBLOCKS. **Codeblocks**. Disponível em: http://www.codeblocks.org/. Acessado em 29 de maio de 2017.

C. **C documentação DevDocs**. Disponível em: http://devdocs.io/c/. Acessado em 29 de maio de 2017.

DEITEL, Harvey; DEITEL, Paul. **C++ Como Programar**. Porto Alegre: Bookman, 2001.

LISTA ENCADEADA. **Lista encadeada (conceito e abordagem), USP**. Disponível em: https://www.ime.usp.br/~pf/algoritmos/aulas/lista.html. Acessado em 27 de maio de 2017.

LAUREANO, Marcos. **Estruturas de Dados com Algoritmos em C**. Rio de Janeiro: Brasport, 2008.

SAVI, Rafael. **Avaliação de Jogos Voltados Para a Disseminação do Conhecimento.** In: Tese (doutorado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Engenharia e Gestão do Conhecimento. Florianópolis, 2011.

TANEMBAUM, Aaron M; LANGSAM, Yedidyah Langsam; AUGENSTEIN, Moshe J. Augenstein. **Estruturas de Dados Usando C**. São Paulo: MARKON Books, 1995.

ZIVIANI, Nivio. **Projeto de Algoritmos com Implementações em Pascal e C.** São Paulo: Editora Pioneira, 1999.