

Engenharia Informática

Estruturas de Dados

2019/2020

Relatório

Recursividade

António Pereira Nº2019141051

David Ferreira Nº2019141281

Data: 10/06/2020

Índice

1	INTRODUÇÃO.....	3
2	TRABALHO REALIZADO	4
2.1	CALCULAR O FACTORIAL DE UM NÚMERO INTEIRO, SABENDO QUE $N! = N \times (N-1) \times (N-2) \times \dots \times 2 \times 1$	4
2.2	CALCULAR O N -ESIMO NÚMERO DE FIBONACCI SABENDO QUE OS <i>NÚMEROS DE FIBONACCI</i> SÃO TERMOS DE UMA SUCESSÃO (COM O MESMO NOME) DEFINIDA POR RECORRÊNCIA DO SEGUINTE MODO: CADA TERMO DA SUCESSÃO É OBTIDO PELA SOMA DOS DOIS TERMOS ANTERIORES. OS PRIMEIROS NÚMEROS DE FIBONACCI SÃO: 1, 1, 2, 3, 5, 8, 13, 21,	5
2.3	INVERTER A ORDEM DOS ELEMENTOS DE UM VECTOR DE NÚMEROS INTEIROS:	7
2.4	ESCREVER UMA SEQUÊNCIA DE ALGARISMOS, DADO UM $N \in \mathbb{N}_0$	9
2.5	LIMPAR UMA LISTA LIGADA (FICHA DE TRABALHO N.º 5, ALÍNEA N) DO EXERCÍCIO 2.), OU SEJA, ELIMINAR TODOS OS ELEMENTOS DE UMA LISTA LIGADA. UMA DAS VERSÕES PODE SER A USADA NA RESOLUÇÃO DA FICHA DE TRABALHO N.º 5.	11
3	CONCLUSÃO	18
4	REFERÊNCIAS.....	19

1 Introdução

Este trabalho é uma introdução a recursividade e como ela funciona. Neste trabalho vamos desenvolver programas utilizando a recursividade que é basicamente quando dentro de uma função é chamada essa mesma função. Também nesta ficha de trabalho é pedido que criemos uma versão iterativa que nada mais é que uma versão normal de um programa.

2 Trabalho realizado

2.1 Calcular o fatorial de um número inteiro, sabendo que $n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1$.

Neste primeiro exercício é pedido para calcular o fatorial de um número usando a recursividade e a versão iterativa. Na versão recursiva criamos uma função em que dentro da função é retornado esta mesma função, que tem como parâmetro o $n-1$, vezes o n que é o número que o utilizador introduz para calcular o fatorial. A função recursiva é a seguinte:

```
int fatorial_recursiva(int n)
{
    if (n <= 0)
        return 1;
    else
        return n * fatorial_recursiva(n-1);
}
```

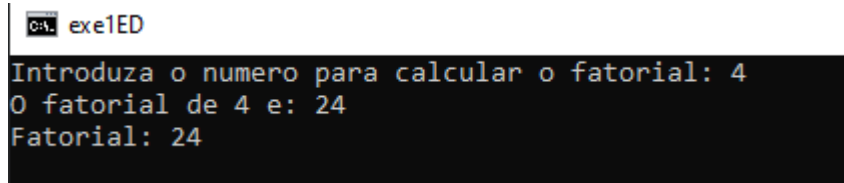
Figura 1- Função recursiva

Já na função iterativa para calcular o fatorial de um número fizemos um for e dentro do for multiplicamos a variável fatorial, que inicialmente é igual a 1, pelo i e a variável fatorial vai ser igual a este valor, deste modo, a variável fatorial vai ficar sempre com o valor da multiplicação, ou seja, vai aumentado o valor e deste modo conseguimos saber o valor do fatorial do número que o utilizador inseriu. Podemos ver a versão iterativa na figura abaixo:

```
int fatorial_iterativa(int n)
{
    int fatorial = 1;
    for(int i=1; i <= n; i++)
        fatorial = fatorial * i;
    return fatorial;
}
```

Figura 2- Versão iterativa para calcular o fatorial

O output deste programa é o seguinte:



```
exe1ED
Introduza o numero para calcular o fatorial: 4
O fatorial de 4 e: 24
Fatorial: 24
```

Figura 3- Interface da função recursiva e da função iterativa

- 2.2 Calcular o n -ésimo número de Fibonacci sabendo que os *Números de Fibonacci* são termos de uma sucessão (com o mesmo nome) definida por recorrência do seguinte modo: cada termo da sucessão é obtido pela soma dos dois termos anteriores. Os primeiros números de Fibonacci são: 1, 1, 2, 3, 5, 8, 13, 21, ...

Neste segundo exercício temos que calcular a série de Fibonacci de um número que é introduzido pelo utilizador. Na função recursiva, se a variável n , que é o número que o utilizador inseriu, for igual a 1 ou 2 a função retorna 1 senão vai retornar esta mesma função, que tem como parâmetro $(n - 1)$, mais outra vez a chamada da mesma função só que desta vez o parâmetro é $(n - 2)$. A função é a seguinte:

```
int fibonacci_rec(int n)
{
    if (n == 2 || n == 1)
        return 1;
    else
        return fibonacci_rec(n-1) + fibonacci_rec(n-2);
}
```

Figura 4- Função recursiva para calcular a série de Fibonacci

Na função iterativa, para calcular a serie de fibonacci fizemos um for e neste for a variável soma vai ser igual a variável a mais a variável b, ou seja, vai ser a soma dos dois últimos termos, pois, a variável a vai ser sempre igual a variável b e o b vai ser igual a soma até o i ser maior que o n. A função iterativa é a seguinte:

```
int fibonacci_ite(int n)
{
    int a = 0;
    int b = 1;
    int soma = 0;
    for(int i=1; i < n; i++)
    {
        soma = a + b;
        a = b;
        b = soma;
    }
    return b;
}
```

Figura 5- Função iterativa

Output das funções recursiva e iterativa:

```
exe2ED
Introduza o numero para calcular a serie de Fibonacci: 4
O numero de Fibonacci de 4 e: 3
O numero de Fibonacci de 4 e: 3
```

Figura 6- Interface das duas funções para calcular a serie de fibonacci

2.3 Inverter a ordem dos elementos de um vetor de numeros inteiros:

Neste exercício, primeiro para inverter a ordem de um vetor tem que se inserir os números do vetor. Para isso, fizemos uma função em que pede ao utilizador para inserir os números do vetor que tem como tamanho o que o utilizador inseriu no início do programa.

Na função recursiva, se o tamanho do vetor for igual ao menor que 1 a função vai retornar 1, pois, não dá para inverter um vetor se tiver só um elemento. Caso seja maior que um, então, o vetor vai ser invertido e para isto, primeiro a variável aux vai ser igual ao valor do elemento da string na posição i, depois, a string na posição i vai ser igual ao valor do elemento na posição j, isto é, a variável string[i] vai ter sempre o valor dos últimos elementos do vetor, e por último a variável string[j] vai ser igual a variável aux que vai ter o valor do elemento na posição i, ou seja, a variável string[j] vai ter sempre o valor dos primeiros elementos do vetor. No fim, vai ser retornado a função que tem como argumentos tam-2 e string +1, ou seja, como j é igual a tam -1 e tam vai ser sempre decrementado duas vezes então o valor de j vai sempre diminuir e deste modo o valor da string na posição j vai ser sempre alterado. Já quando é feito string +1 é para a string andar para a frente. A função é a seguinte:

```
int inverternum_rec(int tam, int string[tam])
{
    int i=0, aux, j;
    j=tam-1;
    if(tam <= 1)
        return 1;
    else
    {
        aux = string[i];
        string[i] = string[j];
        string[j] = aux;
    }
    return inverternum_rec(tam-2, string+1);
}
```

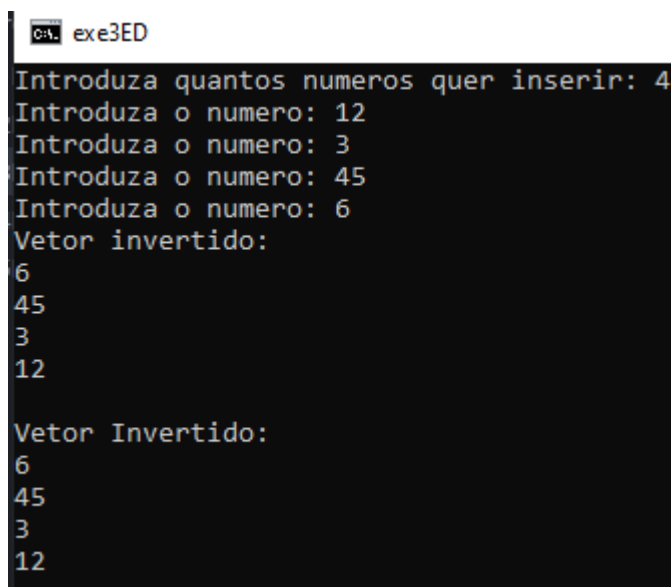
Figura 7- Função para inverter a ordem de um vetor com recursividade

Na função iterativa é realizado um for em que o m vai ser igual a última posição do vetor e enquanto m for maior ou igual a zero vai ser realizado um printf da string na posição m, ou seja, vai ser apresentado os elementos da string na ordem invertida. A função iterativa é a seguinte:

```
void inverternum_ite(int tam, int string[tam])
{
    printf("Vetor invertido: \n");
    for(int m = tam-1; m >= 0; m--)
    {
        printf("%d\n", string[m]);
    }
}
```

Figura 8- Função iterativa para inverter a ordem de um vetor

O output das duas funções é o seguinte:



```
C:\ exe3ED
Introduza quantos numeros quer inserir: 4
Introduza o numero: 12
Introduza o numero: 3
Introduza o numero: 45
Introduza o numero: 6
Vetor invertido:
6
45
3
12

Vetor Invertido:
6
45
3
12
```

Figura 9- Vetor invertido

2.4 Escrever uma sequência de algarismos, dado um $n \in \mathbb{N}$

Para este exercício, foi-nos pedido para escrevermos uma sequência de algarismos dado um n introduzido pelo utilizador, ou seja, se o utilizador inserir o número 4 como n a função tem que apresentar 4, 3, 2, 1, 2, 3, 4.

Na função recursiva, se o n for igual vai apresentar o 1 se for maior que 1 então vai ser realizado um printf do valor do n , depois a função vai retornar $n - 1$ e vai ser realizado outra vez um printf até o n ser 1. Quando o n for 1 então o printf a seguir a chamada da função vai apresentar os valores do n que ficaram esquecidos, isto é, se o n for igual 4 vai escrever o 2, 3, 4 pois o 4, 3, 2, 1 já foram escritos no primeiro printf. A função recursiva é a seguinte:

```
void sequencia_rec(int n)
{
    if(n == 1)
        printf("%d", n);
    else
    {
        printf("%d", n);
        sequencia_rec(n-1);
        printf("%d", n);
    }
}
```

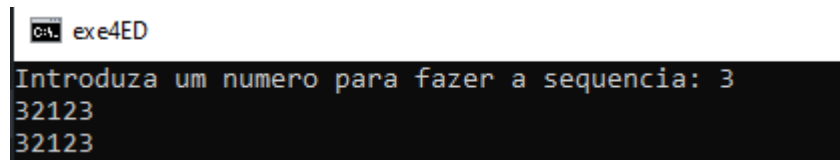
Figura 10- Função recursiva para escrever uma sequência

Para escrever a sequência de algarismos na função iterativa realizamos dois `while`s. No primeiro enquanto o `n` for maior que 1, primeiro vai ser escrito o valor de `n` e depois o `n` é decrementado, ou seja, se o `n` for 4 passa a 3 e assim sucessivamente até o `n` ser igual a 1 que é quando o `while` para. Já no segundo `while` é onde vai ser escrita a segunda parte da sequência, ou seja, enquanto `n` for menor ou igual a `m`, variável declarado no início da função e que é igual a `n`, vai ser apresentado o valor de `n` e depois incrementado o `n`, ou seja, como sabemos que o `n` é 1 depois do primeiro `while` então o `n` vai passar a 2 depois a 3 e por último a 4, caso o utilizador no início desse o valor de 4 ao `n`. Na figura abaixo podemos ver a função iterativa:

```
void sequencia_ite(int n)
{
    int m = n;
    while(n > 1)
    {
        printf("%d", n);
        n--;
    }
    while(n <= m)
    {
        printf("%d", n);
        n++;
    }
}
```

Figura 11- Função iterativa

O output destas duas funções é o seguinte:



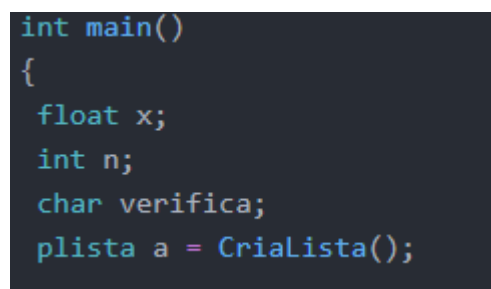
```

C:\ exe4ED
Introduza um numero para fazer a sequencia: 3
32123
32123
  
```

Figura 12- Sequencia de algarismos caso o n seja 3

2.5 Limpar uma lista ligada (Ficha de Trabalho n.º 5, alínea n) do exercício 2.), ou seja, eliminar todos os elementos de uma lista ligada. Uma das versões pode ser a usada na resolução da Ficha de Trabalho n.º 5.

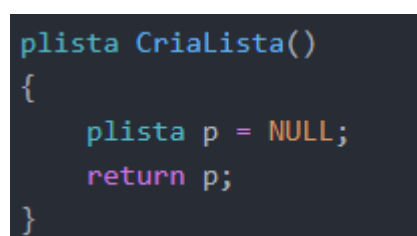
Neste último exercício, foi-nos pedido que limpássemos uma lista ligada, isto é, eliminar todos os elementos de uma lista. Para fazer isto, primeiro tivemos que criar uma lista vazia que é basicamente criar uma variável do tipo plista e igualá-la a NULL. A função não tem argumentos, pois no main foi chamada da seguinte forma:



```

int main()
{
    float x;
    int n;
    char verifica;
    plista a = CriaLista();
  
```

Figura 14- Chamada da função CriaLista



```

plista CriaLista()
{
    plista p = NULL;
    return p;
}
  
```

Figura 13- Função para criar lista

Depois de criada a lista tivemos que fazer uma função para inserir os elementos na lista. Nesta função, pedimos ao utilizador o número de elementos da lista e realizamos um for onde vai ser chamada a função JuntaFim, que é a função para inserir um elemento na lista. Na função JuntaFim, caso a lista esteja vazia, o primeiro elemento será o último. Assim, nesse caso basta igualar o nó ao primeiro elemento da lista e fazer com que o ponteiro passe a apontar para NULL. Caso a lista não esteja vazia, a, temos que percorrer a lista até encontrarmos o último elemento e igualar este último elemento ao que nós queremos colocar lá e passar o valor do último elemento para o lado esquerdo. No entanto, para o elemento ser, sem dúvida, o último, o ponteiro terá que apontar para NULL. Para não perdermos a lista, criámos um ponteiro auxiliar que aponta para o primeiro elemento da lista. A função para inserir um elemento na lista:

```
void JuntaFim(float x, plista * lst)
{
    plista no = (plista) malloc(sizeof(listano));
    plista paux = *lst;
    if (no == NULL)
        printf("\nErro, nao ha memoria disponivel.");
    else {
        no->valor = x;
        no->prox = NULL;
        if (*lst == NULL) {
            *lst = no;
        } else {
            while (paux->prox != NULL) {
                paux = paux->prox;
            }
            paux->prox = no;
        }
    }
}
```

Figura 15- Função JuntaFim que serve para inserir um elemento na lista

Função para inserir mais que um elemento na lista:

```
void MontaLista(plista * lst, int max)
{
    float valor;
    for(int i=1; i<=max; i++)
    {
        printf("\nInsira o valor que deseja inserir na posicao %d da lista: ", i);
        scanf("%f", &valor);
        JuntaFim(valor, lst);
    }
}
```

Figura 16- Função que serve para inserir mais que um elemento na lista

Output da função MontaLista:

```
Insira o numero de elementos da lista: 4
Insira o valor que deseja inserir na posicao 1 da lista: 12
Insira o valor que deseja inserir na posicao 2 da lista: 3
Insira o valor que deseja inserir na posicao 3 da lista: 45
Insira o valor que deseja inserir na posicao 4 da lista: 67
[12.0, 3.0, 45.0, 67.0, ]
```

Figura 17- Elementos da lista

Para apresentar os elementos da lista também criamos uma função. Esta função permite o utilizador e nós ver os elementos da lista e ver se tudo correu bem. A função é a seguinte:

```
void Mostralista(plista lst) {
    if (ListaVazia(lst)=='y')
        printf("\nA lista esta vazia.");
    else
    {
        printf("\n");
        printf("[");
        do {
            printf("%.1f, ", lst->valor);
            lst = lst->prox;
        } while (lst != NULL);
        printf("\b ]\n");
    }
}
```

Figura 18- Função para apresentar os elementos de uma lista

Output desta função:

```
[12.0, 3.0, 45.0, 67.0, ]
```

Figura 19- Interface da função MostraLista

Após criarmos a função para inserir os elementos numa lista, fizemos duas funções, uma iterativa outra recursiva, para limpar a lista.

Na função iterativa, fizemos um while e dentro deste é chamada a função RemoveUltimo_Ite, que é a função, que serve para remover um elemento da lista. Nesta função, utilizamos dois ponteiros em que o ponteiro paux aponta para o início da lista e o ponteiro aux é uma lista vazia. Deste modo, caso a lista não esteja vazia, o ponteiro aux irá andar “atrás” do ponteiro paux, de modo a que, quando é encontrado o último elemento, se possa eliminar o paux, utilizando o free(paux) e, caso a lista tenha mais que um elemento, o aux passa a apontar para NULL, ou seja, passa a ser o último elemento. Caso só tenha um elemento, a lista passa a ser vazia. A função para remover um elemento da lista é a seguinte:

```
void RemoveUltimo_Ite(plista * lst)
{
    plista paux = *lst;
    plista aux = NULL;
    while(paux->prox!=NULL)
    {
        aux = paux;
        paux = paux->prox;
    }
    free(paux);
    if(aux!=NULL)
        aux->prox = NULL;
    else
        *lst = NULL;
}
```

Figura 20- Função que remove um elemento da lista

Função iterativa para limpar a lista:

```
void limpalista_Ite(plista * lst)
{
    while(*lst!=NULL)
        RemoveUltimo_Ite(lst);
}
```

Figura 21- Função que serve para limpar lista

Já na versão recursiva, primeiro criamos um ponteiro auxiliar que aponta para o início da lista, depois, fizemos um while e dentro deste a lista vai andar para a frente através da instrução `(*lst = (*lst)->prox)`, vai ser eliminado o valor de `paux` através do `free` e ainda dentro do while vai ser chamada esta mesma função que tem como argumento a lista. A função recursiva é a seguinte:

```
void Limpalista_Rec(plista * lst)
{
    plista paux = *lst;
    while(*lst != NULL)
    {
        *lst = (*lst)->prox;
        free(paux);
        Limpalista_Rec(lst);
    }
}
```

Figura 22- Função recursiva para eliminar os elementos da lista

Output das duas funções para eliminar os elementos de uma lista:

```
A lista esta vazia.
Insira o numero de elementos da lista: 4

Insira o valor que deseja inserir na posicao 1 da lista: 12
Insira o valor que deseja inserir na posicao 2 da lista: 3
Insira o valor que deseja inserir na posicao 3 da lista: 45
Insira o valor que deseja inserir na posicao 4 da lista: 67

[12.0, 3.0, 45.0, 67.0, ]
A lista esta vazia.
```

Figura 23- Output da função recursiva


```
Insira o numero de elementos da lista: 3
Insira o valor que deseja inserir na posicao 1 da lista: 12
Insira o valor que deseja inserir na posicao 2 da lista: 3
Insira o valor que deseja inserir na posicao 3 da lista: 4
[12.0, 3.0, 4.0, ]
A lista esta vazia.
```

Figura 24- Output da função iterativa

3 Conclusão

Concluindo, este trabalho foi interessante e útil para aprendermos a utilizar a recursividade que ao início parecia confuso, mas foi simples de perceber. Assim, achamos que fizemos um bom trabalho e que nos deu novos conhecimentos sobre a linguagem c.

4 Referências

<https://www.vivaolinux.com.br/script/Fatorial-Recursoivo>

<https://pt.stackoverflow.com/questions/177742/impress%C3%A3o-fibonacci-recursivo>