Curso de Programação em C

Introdução à Programação

Lucas Maximiliano de Oliveira e Silva

Colégio Visconde de Porto Seguro

28 de setembro de 2022

Sumário

Revisão

I/O:

- Input → scanf();
- Output → printf();
- "Format-specifiers":
  - %c: char;
  - %x: hexadecimal;
  - %i : int (ou %d);
  - %.**n**lf: double.
- \n: Pular linha.

Loops:

```c
for(int i=0; i<2; i++) {
    printf("i = %d\n", i);
}
```

```c
int i = 2;
while(i!=0) {
    printf("i = %d\n", i);
    i--;
}
```

```c
int i = 0;
do {
    printf("i = %d\n", i);
    i--;
} while(i>0);
```

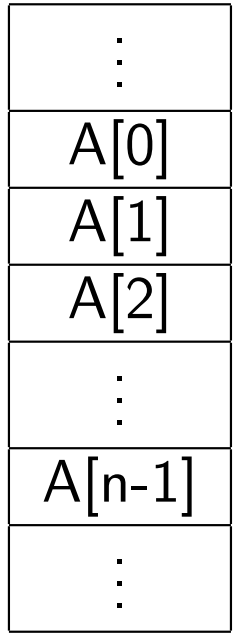
If-Else: Decidir entre **2** estados;

Switch: Decidir entre **vários** estados.

Matrizes

Matriz: Matrizes (*Array*) armazenam informação em blocos de memória.

- Tamanho n do Array é fixo;
- Index inicial é 0;
- Nome do Array == endereço do 1° elemento;
- Acessar elementos individuas com A[i], sendo "i"o índice do elemento, ou *ponteiros*;



Exemplo 1

```c
#include <stdio.h>
int main() {
    //Declarar Array A[] de 5 elementos:
    int A[5];
    //Inicializar Array:
    for(int k=0; k<5; k++) {
        A[k] = k/2;
        printf("A[%i] = %i\n", k, A[k]);
    }
    return 0;
}
```

Output

```
A[0] = 0
A[1] = 0
A[2] = 1
A[3] = 1
A[4] = 2
```

Exemplo 2

Matrizes também podem ter mais de uma dimensão. Por exemplo tabelas (2 dimensões: horizontal e vertical.)

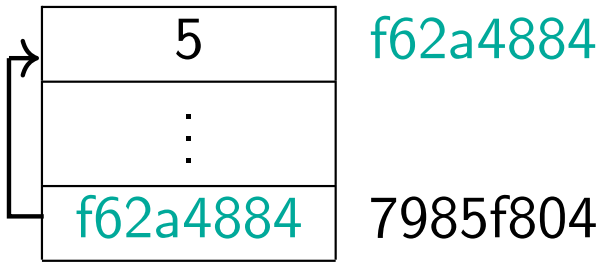
```
int A[m][n];      //m linhas e n colunas
```

A[0][0]	A[0][1]	A[0][2]	...	A[0][n-1]
A[1][0]	A[1][1]	A[1][2]	...	A[1][n-1]
⋮	⋮	⋮	⋱	⋮
A[m-1][0]	A[m-1][1]	A[m-1][2]	...	A[m-1][n-1]

Ponteiros

Ponteiro: Ponteiros são variáveis especiais, cuja informação que armazenam é o *endereço* de uma outra estrutura.

```c
#include <stdio.h>
int main() {
    int a = 5;
    //Declarar um ponteiro:
    int* ptr;
    //Inicializar um ponteiro:
    ptr = &a;
    printf("Endereco a:\n%x",ptr);
    printf("Valor a:\n%i",*ptr);
    return 0;
}
```



Output

```
Endereco a:
f62a4884
Valor a:
5
```

Funções

Função: Uma função é um trecho de código com um objetivo bastante específico.

- Objetivo: Compartimentar código;
  - Menos erros (localidade);
  - Mais legível;
- Declaração (funcoes.h) $\rightarrow$ Definição (funcoes.c) $\rightarrow$ Uso (main.c);

Estrutura de uma Função

```
returnType functionName(argType arg1, argType arg2) {
    //...
    return result
}
```

- returnType: void, int, double, bool, etc.;
- arg1, arg2: Argumentos → input;
- return: Comando para retornar resultado → output;

Exemplo

main.c

```
1  #include <stdio.h>
2  //Tudo no mesmo diretorio
3  #include "funcoes.h"
4  int main() {
5      //Tamanho especificado
6      //pela lista:
7      int v[] = {0,1,7,3,9};
8      int w[] = {2,2,5,6,1};
9      int r = dotPrd(v,w,5);
10     printf("r = %i", r);
11     return 0;
12 }
```

Output

```
r = 64
```

funcoes.h

```
1  #ifndef _FUNCOES_H_    //Header Guard
2  #define _FUNCOES_H_
3  int dotPrd(int v1[], int v2[], int n);   //Declarar:
4  #endif
```

funcoes.c

```
1  #include "funcoes.h"
2  int dotPrd(int v1[], int v2[], int n) {   //Definir:
3      int res = 0;
4      for(int i=0; i<n; i++) {
5          res += v1[i] * v2[i];
6      }
7      return res;
8  }
```

Local VS. Global

Global: Variável acessível a todo programa;

Local: Variável acessível somente a partes do programa;

Exemplo em Pseudo-código:

```
i = a                    //global
Foo1() {
    Foo2() {
        i = b    //local
        print("print 1 : " i)
    }
    print("print 2 : " i)
}
```

Output

```
print 1 : b
print 2 : a
```

Recursão

Recursão: Quando uma função chama a si mesma. Cada "chamada" tem parâmetros diferentes e são conhecidas por *Stackframes*.

- Funções recursivas sempre possuem um caso não-recursivo, para evitar um ciclo infinito de repetições;
- Código bastante limpo, curto e elegante (mas nem sempre eficiente...);

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=3 return fib(2)+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=2 return fib(1)+fib(0)
n=3 return fib(2)+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=1	return 1
n=2	return fib(1)+fib(0)
n=3	return fib(2)+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=0	return 0
n=2	return 1+fib(0)
n=3	return fib(2)+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=2 return 1+0
n=3 return fib(2)+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=3 return 1+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=1 return 1
n=3 return 1+fib(1)

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=3 return 1+1

Exemplo

Calcular os n°s de Fibonacci: 0, 1, 1, **2**, 3, 5, 8, ...

```
1  int fib(int n) {
2      if(n==0)
3          return 0;
4      if(n==1)
5          return 1;
6      else
7          return fib(n-1)
8                 + fib(n-2);
9  }
```

Stack para calcular fib(3):

n=3 return 1+1

Como melhorar? Caching → calcular fib(1) apenas x1