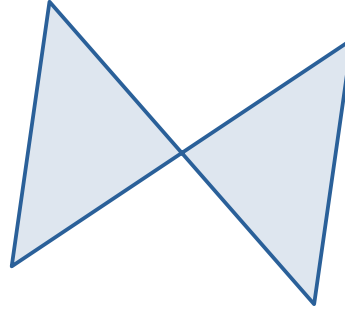
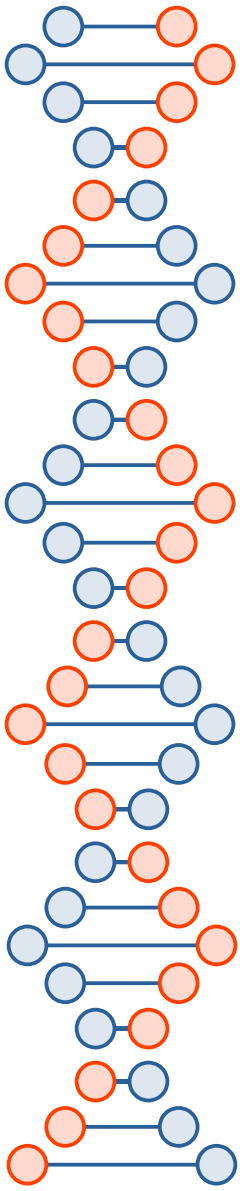
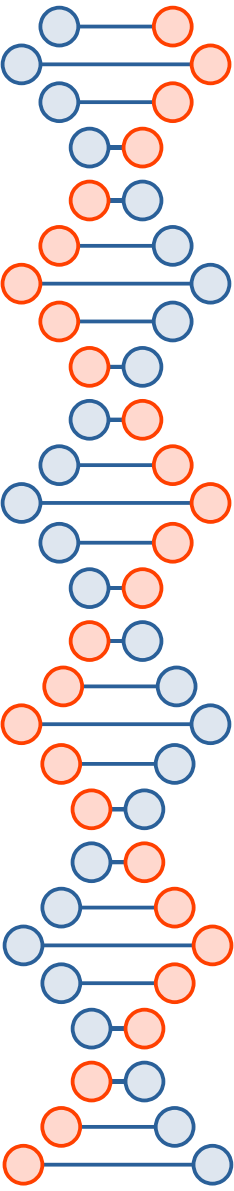


Fundamentos da linguagem C



Formatos



| | |
|----------|-----------------------------------------------------------|
| %c | = char (caracter) 1 byte |
| %i ou %d | = int (número inteiro) 4 bytes |
| %s | = string (vetor de caracter) 1 byte * nCaracteres |
| %f | = float (número real, tem casas decimais) 4 bytes |
| %u | = unsigned (número inteiro sem sinal, 0 ou maior) 4 bytes |
| %l | = long (pode vir antes de d ou i, ocupa o dobro de bytes) |
| %ll | = long long (análogo ao long, ocupa o dobro de bytes) |

Estruturas Condicionais

Cláusulas

- if (se)
- else (senão)
- else if (senão, se)
- switch (alterna)

Em casos que o = vem acompanhado de outro símbolo, ele sempre vem à DIREITA. Seja em comparações (\geq) ou em operações aritméticas ($+=$) que utilizam o termo à esquerda.

Condições

- $<$ e $>$ (menor e maior que)
- \leq e \geq (menor ou igual e maior ou igual)
- \neq (diferente de)
- ! **NÃO/NOT** lógico
- \parallel **OU/OR** lógico, $\&\&$ **E/AND** lógico

if() recebe como argumento uma condição, como previsto na tabela acima. Exemplos:

`if(numero1 > numero2)` (se numero1 for maior que numero 2)

`if(caracter == 'a' || caracter == 'b')` (se caracter for 'a' ou 'b')

`if(arquivo != NULL)` equivale a `if(arquivo)`,
da mesma forma que `if(!verdadeiro)` equivale a `if(verdadeiro == 0)`

A escolha é de quem usa, mas vale entender a semelhança funcional.

Estruturas Condicionais (switch)

```
#include <stdio.h>
#define SUCESSO      (0)

int main(int argc, char ** argv){
    int escolha;
    char lixo;

    printf("Qual opcao desejas?\n"
           "1 - Ser feliz\n"
           "2 - Ser rico\n"
           "3 - Ser famoso\n"
           "4 - Ser amad@?\n");
    scanf("%d%c", &escolha, &lixo);

    switch(escolha){ // faz a verificacao das condicoes dentro da variavel 'escolha'
        case 1:      // abre a condicao
            printf("(:"); // codigo da condicao
            break;     // encerra a condicao, analogo ao }
        case 2:
            printf("$$$$");
            break;
        case 3:
            printf("ME DA UM AUTOGRAFO??!");
            break;
        case 4:
            printf("Nossa bb kkkkkk saudade do que a gente nao viveu ainda di vdd 9vinha");
            break;
        default:
            printf("Opcao invalida"); // default compreende qualquer opcao EXCETO as que voce ja estabeleceu
            break;
    }

    return SUCESSO;
}
```

Estruturas Condicionais (else + if)

Caso precisemos trabalhar com condições lógicas e comparativas, entra if, else e else if.

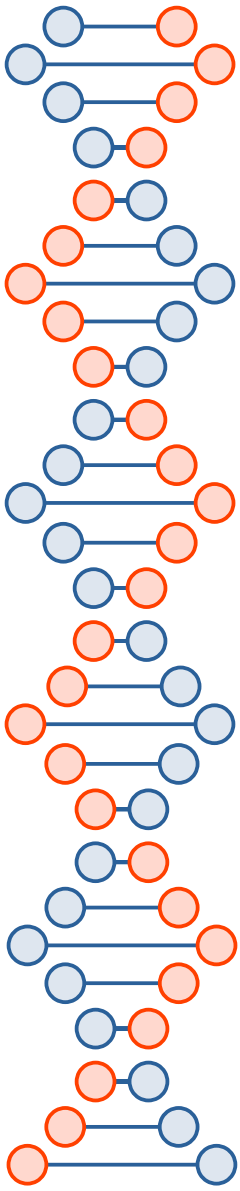
```
#include <stdio.h>
#define SUCESSO      (0)

int main(int argc, char** argv) {
    int n;
    char lixo;

    printf("Insira seu numero: ");
    scanf("%d%c", &n, &lixo);

    if (n > 0) { // SE numero for maior que 0
        printf("\n0 numero dado eh inteiro e POSITIVO.");
    }
    else if (n < 0) { // SENAO, SE for maior que 0, e sim menor que 0
        printf("\n0 numero dado eh inteiro e NEGATIVO.");
    }
    else { // SENAO, eh 0
        printf("\nProvavelmente, o numero dado eh 0. Acertei?");
    }

    return SUCESSO;
}
```



Funções

Funções são declaradas como variáveis. Elas tem tipo, nome único e tamanho. São interpretadas normalmente na ordem que está disposta no código, isto é, para chamar uma função, ela precisa vir acima. O diferencial da função é os parênteses (), que compreendem os parâmetros e as chaves {}, que compreendem o seu código.

```
#include <stdio.h>
#define SUCESSO      (0)

int main(int argc, char** argv) {

    return SUCESSO;
}
```

*int main(int argc, char ** argv)* é um exemplo de função. Trabalhamos todo o tempo sem nem notar, mas o main tem dois parâmetros:

- 1) *int argc*
- 2) *char** argv*

Parâmetros são variáveis exclusivas da função, cujos valores são passados na CHAMADA da função. Estes verdadeiros valores passados são os argumentos. No caso exclusivo do main(), a chamada é implícita.

Funções

```
#include <stdio.h>
#define SUCESSO      (0)

int funcao_soma(int termo_a, int termo_b) {
    return termo_a + termo_b;
}

int main(int argc, char** argv) {
    printf("%d", funcao_soma(5, 15));

    return SUCESSO;
}
```

DECLARAÇÃO DA FUNÇÃO

Os parâmetros, são duas variáveis inteiras, "termo_a" e "termo_b". Não importa qual valor você passar como argumento, dentro da função ele o irá tratar pelo nome do parâmetro declarado.

CHAMADA DA FUNÇÃO

São passados os argumentos 5 para "termo_a" e 15 para "termo_b".

As funções que tem tipo diferente de 'void', obrigatoriamente retornam um valor. Retornar no sentido de assumir um ÚNICO valor ou ponteiro, que a função irá "devolver" após sua execução, e você pode guardar em uma variável. Esse valor irá depender dos argumentos que você forneceu. Em uma função matemática, como se retornasse y em função dos argumentos x.

saída:
20