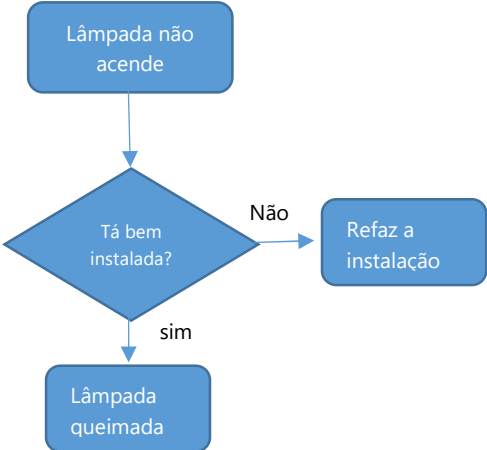


Revisitando

Este material é uma visita rápida por alguns conceitos e práticas abordados na disciplina de Fundamentos de Programação (FUP). Vamos lá?

O primeiro conceito importante seria, o que é um **algoritmo**?

Podemos defini-lo como um conjunto de passos finitos para executar uma tarefa. Ou seja, é necessário descrever o início, meio e fim no processo de execução de uma tarefa. E esse processo pode ser representado de diversas maneiras, como exemplo:

Fluxograma	Narrativa
 <pre>graph TD; A[Lâmpada não acende] --> B{Tá bem instalada?}; B -- Não --> C[Refaz a instalação]; B -- sim --> D[Lâmpada queimada];</pre>	<ol style="list-style-type: none">1. Verifica que a lâmpada não acende2. Verifica se ela está bem instalada3. Se sim a lâmpada está queimada4. Se não, refaz a instalação

Tudo certo por enquanto? Espero que sim!

Bom, na disciplina de FUP vimos que para o computador compreender os passos que descrevemos para realizar uma tarefa é necessário fazer uso de uma linguagem definida e estruturada, e que para isso, nesta disciplina e no início do nosso curso, fazemos uso da **Linguagem de Programação C**. Uma linguagem de fácil compreensão e simples de ser utilizada.

E ao fazer uso dessa linguagem é preciso compreender **alguns elementos**:

Identificadores

São utilizados para nomear variáveis, constantes, nomes de funções e outros elementos. Porém, há regras para sua constituição, um identificador em C, deve:

- Ter como primeiro símbolo uma letra, os próximos símbolos podem ser letras, dígitos ou subscrito/underline (_).

- não conter caracteres especiais, por exemplo: \$, %, &, *, entre outros.
- não ser igual a palavras reservadas da linguagem C, tais como: int, else, do, void, entre outras.

Tipo

Os dados manipulados nos programas em C devem possuir tipos. E para nossos primeiros passos na disciplina conhecemos alguns deles, demoninados primitivos ou básicos:

Tipo	Nome	Tamanho (em bits)
char	Caractere	8
int	Números inteiro	16
float	Real, ou Ponto flutuante	32
double	Duplo, representa números reais maiores	64
void	Vazio	-
bool	Lógico ou booleano (verdadeiro ou falso)	8

No decorrer da disciplina conheceremos outros tipos de dados, vamos poder renomea-los e até criar nossos próprios tipos.

Declaração de Variáveis

Variável é um espaço na memória virtual para alocação de dados. Em um programa C uma variável deve ser "criada" (alocada) apenas uma única vez, respeitando a regra:

tipo identificador;

Ou seja, em C todas as variáveis possuem tipos, dizemos então que a linguagem é tipada ou tipificada.

Constantes

Constantes também são variáveis, também possuem tipos, porém seus dados não podem ser modificados após sua alocação. Para diferenciar das demais, uma variável constante é declarada através de uma diretiva (#):

```
#define PI 3.14
```

```
#define nome_mãe "Laura"
```

Biblioteca

Bom, falando em diretivas, lembramos de outra, a `#include` que possui a responsabilidade de incluir/anexar bibliotecas ao nosso código fonte. As bibliotecas são arquivos que contêm pequenos programas, variáveis e constantes definidos, que podemos fazer uso no processo de desenvolvimento do nosso programa. Já conhecemos algumas delas:

- **#include <stdio.h>** : Biblioteca de entrada e saída de informações. Com ela fazemos uso das funções "printf" e "scanf"
- **#include <stdlib.h>** : Biblioteca de uso geral. Com ela podemos fazer uso de algumas funções de conversão.
- **#include <math.h>** : Biblioteca com funções matemáticas e constantes.

Bom, visto esses elementos, podemos lembrar a **estrutura de um programa em C**?

Vamos lá! Um programa em C é a reunião de uma ou mais funções, em que a estrutura básica é composta por:

```
#include <stdio.h>
#define x 10
main() {
    int a;
    printf("Digite um numero inteiro: ");
    scanf("%d", &a);
    if ( a < x)
        printf("A = %d eh menor que X = %d\n", a, x);
    else
        printf("A = %d nao eh menor que X = %d\n", a, x);
}
```

} Diretivas do pré-processador

} Cabeçalho da função principal

} Declaração de variáveis e comandos da função "main".

Pré-processamento

É um programa que realiza a substituição dos comandos que contêm as diretivas (#) por seus códigos. Desta forma, ao fazer uso do comando `#include <stdio.h>`, o pré-processador

incluirá todo o código da biblioteca de entrada e saída no programa que compara dois números, permitindo assim o uso das funções printf e scanf.

Programa Principal

Em C só é permitido uma função principal, que "orquestra" ou "coordena" todo o fluxo da execução do programa. Esta função é nomeada por **main**.

Estamos quase terminando nossa "revisitação", aos conceitos iniciais da disciplina. Falta pouco.

Que tal lembrarmos os **comandos de entrada e saída**?

Printf e Scanf

São funções pertencentes a biblioteca **stdio.h**, cuja a função é ler (scanf) e escrever (printf) valores. Mas só existem essas funções para leitura e escrita de dados? A resposta é não, mas por hora está é a primeira que conhecemos, com o passar do tempo e amadurecimento na disciplina, vamos conhecer as demais.

Como utiliza-las?

O primeiro elemento (argumento) para a função printf é obrigatório e corresponde a um conjunto de caracteres, apresentado entre aspas duplas "*por exemplo, este aqui*". A mesma função pode ter outro argumento, que corresponde as variáveis cujo o valor será impresso na frase que estiver entre aspas. Por exemplo:

```
main(){
    int n1=5;
    float n2=4.8;
    char letra='r';
    printf("As variáveis são %d, %f, %c respectivamente .\n", n1, n2, letra);
}
```

Desta forma, o símbolo % corresponde ao formatador do tipo de variável. Bom, lembra aqueles tipos simples de dados que abordamos no início desse documento? Para alguns deles temos um formatador específico:

Tipo	Formatação nas funções printf() e scanf()
char	%c
int	%i ou %d
float	%f
double	%lf

void	-
bool	-

E ao utilizar este formatador convertemos os valores armazenados em valores a serem impressos.

Bom, com o scanf as regras de uso são quase semelhantes. A diferença é que possui dois argumentos obrigatórios, o primeiro corresponde ao formato da variável e o segundo corresponde ao endereço de memória em que a variável foi alocada na memória, para este último, fazemos uso do comando (&). Como o exemplo:

```
main() {  
    int a;  
    printf("Digite um numero inteiro: ");  
    scanf("%d", &a);  
}
```

Bom, tudo certo? Chegamos ao fim da revisão dos conceitos iniciais da disciplina. No próximo material trabalharemos os comandos condicionais.