{introcomp}

# Working 04 : Conceitos Básicos I

**Objetivos:**

- Dominar os conceitos básicos da linguagem de programação C;
- Aprender a utilizar o compilador, identificando os erros de sintaxe do código fonte;

**Prazo de Envio:** segunda, 12/11, 18:00h.

# 1 CONSTANTES

Em algumas situações surge a necessidade de se utilizar um determinado valor constante em diversas partes do código. Para simplificar a alteração dessas constantes e facilitar a leitura do código, é possível definir um nome significativo para elas de maneira simples, por meio de uma estrutura específica.
Essa estrutura é iniciada pela diretiva #define, seguida pelo identificador da constante e pelo seu respectivo valor, todos separados por um espaço em branco. #define <identificador><valor> Observe que não é acrescentado o ';' no final. Geralmente, para diferenciar as constantes no código, opta-se por escrever seus nomes com todas as letras maiúsculas.
Exemplos:

```
1 #define TRUE 1
  #define FALSE 0 //agora, a palavra FALSE esta vinculada ao valor 0
```

# 2 EXPRESSÕES

As variáveis e constantes podem ser combinadas com os operadores associados a cada tipo de dados, gerando expressões.

## 2.1 Expressões Aritméticas

A expressão aritmética é aquela cujos operadores e operandos são de tipos numéricos (int ou float, por exemplo). Nas expressões aritméticas, é guardada sempre a seguinte relação de prioridade:

1. Multiplicação, divisão e resto de divisão (*,/,%, respectivamente);
2. Adição e subtração (+,-, respectivamente).

Além das operações básicas citadas acima, é possível usar, nas expressões aritméticas, outras operações, bastante comuns na matemática, definidas dentro da linguagem na forma de funções. A sintaxe geral para o uso dessas funções é a seguinte: <nome da função>(<valor>);
Algumas das funções matemáticas existentes são: sin (seno de um ângulo em radianos), cos (cosseno), pow (potenciação), sqrt (raiz quadrada de um número) etc. Para saber mais, verifique no livro texto.
Para a utilização dessas funções é necessário a inclusão da biblioteca math.h, além de ser necessário compilar o código adicionando a tag '-lm'.
Exemplo:

```
gcc programa.c -o prog -lm
```

# 3 COMANDOS DE ENTRADA DE DADOS

O comando de entrada de dados serve para captar do usuário do programa um ou mais valores necessários para a execução das tarefas. Na verdade, o que se faz é ler os dados de uma fonte externa, normalmente do teclado, para depois processar os dados e obter uma saída. Por meio desse comando de leitura de dados, uma ou mais variáveis podem ter seus valores lidos durante a execução do programa. Essa característica permite criar programas mais flexíveis.
Na linguagem C, a sintaxe para o uso do comando é a seguinte:

```
scanf("<formato1><formato2> ... <formatoN>", &var1, &var2, ..., &varN);
```

Essa estrutura é dividida em duas partes distintas. A primeira, colocada entre aspas duplas, contém os formatos, os quais são relacionados diretamente com os tipos das variáveis a serem lidas. A segunda parte é uma lista dos nomes dessas variáveis, com uma relação direta entre a posição nessa lista e o respectivo formato descrito na primeira parte do comando. Conforme mostra a descrição da sintaxe, os nomes das variáveis devem ser precedidos pelo caractere '&'.

Os tipos de formatos mais utilizados são:

- %c – lê um único caractere;
- %d – lê um inteiro;
- %lf – lê um número em ponto flutuante (real).

É importante lembrar que o comando scanf é oriundo da biblioteca stdio.h.
Exemplos:

```
double a, n1;
int b, n2;
char c = 'b';
scanf ("%d %lf",&b,&a);
scanf ("%d %lf",&n1,&n2); //observe que esta incorreto, pois %d e o formato para numeros
    inteiros sendo que n1 e do tipo float. Assim como para o n2
```

# 4 COMANDO DE SAÍDA DE DADOS

Até agora, os exemplos de códigos e os conceitos estudados não mostravam como os programas comunicavam seus resultados ao usuário no final do programa, as variáveis armazenavam os resultados, mas o usuário não tinha como acessá-los. De fato, esse é o papel do comando de saída de dados.
O comando de saída de dados serve para escrever mensagens e exibir valores de expressões, proporcionando mais informação e legibilidade tanto para o usuário quanto para o próprio programador.
Na linguagem C, a sintaxe para o uso do comando de saída é mostrada a seguir.

```
printf("<formato1><formato2> ... <formatoN>", exp1, exp2, ..., expN);
```

Os formatos são os mesmos utilizados no scanf. Além disso, há um especficador '\n' que pula uma linha, evitando assim que o especificador do terminal e a saída do programa fiquem na mesma linha.
Exemplos de uso do printf:

```
double altura, peso;
printf ("Digite sua altura e peso:");
scanf ("%lf %lf",&altura", &peso);
printf ("Sua altura e peso sao: %f m %f kg\n",altura,peso);
```

Para especificar limites para casas decimais em números reais, basta acrescentar entre o % e lf um ponto (.), mais a quantidade de casas decimais que você deseja utilizar.
Exemplo:

```
double altura, peso; // supondo a leitura de 2.1234 e 4.322123
scanf ("%lf %lf",&altura, &peso);
printf ("Sua altura e peso sao: %.2f m %.4f kg\n",altura,peso); //saida: 2.12 e 4.3221
```

# 5 COMPILANDO E EXECUTANDO

O compilador é um programa que transforma o código-fonte escrito em linguagem de alto nível, para um executável em linguagem de máquina.
No sistema operacional Linux, para compilar um código-fonte utilize os seguintes comandos:

```
gcc <codigo_fonte>.c ou gcc  <codigo_fonte> -o <nome_do_executavel>
```

Lembrando que utilizando o primeiro comando, o compilador irá criar um executável com o nome padrão a.out. Caso queira dar um nome para o executável, utilize o comando com o parâmetro -o <nome_do_executavel>.
Para executar, basta utilizar o seguinte comando:

```
./<nome_do_executavel>
```

# Praticando

Agora vamos praticar! Para todos os praticando em que é pedido que se escreva um programa, escreva o código do seu programa e nos envie o .c (código fonte) correspondente.

1. Faça um programa que leia 2 números inteiros do teclado, e imprima a soma deles.

```c
#include <stdio.h>
int main()
{
  int soma, num1, num2; //declaracao das variaveis
  //utilize a funcao scanf para ler os numeros do teclado
  //some os numeros lidos do teclado
  //imprima o resultado da soma com o comando printf
  return 0;
}
```

2. Crie um programa em C que leia duas variaveis inteiras do teclado e imprima o conteudo delas.

3. Passe para C os algoritmos a seguir:

   (a) Algoritmo 1

```
inicio {Algoritmo para o calculo da Media Final }
real : P1,P2,P3,MF;
imprima("Entre com o valor das Parciais P1, P2, P3");
leia (P1,P2,P3);
MF <- (P1+P2+P3)/3.0;
imprima ("A Media Final e: ", MF);
fim
```

   (b) Algoritmo 2

```
inicio {Algoritmo para o calculo da area de um retangulo}
real : AREA, L1, L2;
imprima("Entre com o valor da Altura e da Largura");
leia (L1, L2);
AREA <- L1*L2;
imprima ("A area do retangulo de lados L1 e L2 e: AREA");
fim
```

(c) Algoritmo 3

```
inicio {Algoritmo para a troca dos valores de duas variaveis}
inteiro : A, B, AUX;
imprima("Entre com o valor de A e de B");
leia (A, B);
AUX <- A;
A <- B;
B <- AUX;
imprima ("A e B");
fim
```

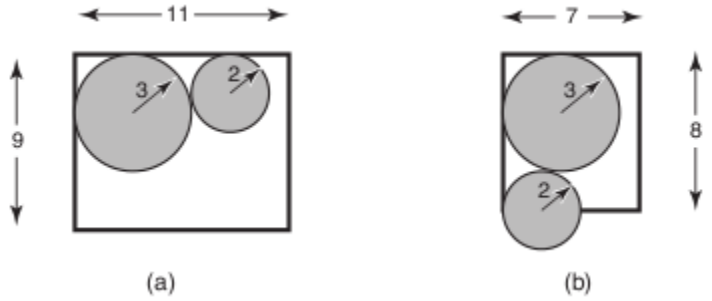
4. Sobre o seguinte codigo mostrado abaixo, aponte sete erros identificando a linha e qual erro em específico foi cometido.

```
#include <std.h>
#define PI 3.14;
int main()
{
    double Area,raio;
    printf("Entre com o valor do raio da circunferencia:);
    scanf("%f",raio);
    area=pi*raio*raio;
    printf("A area da circunferencia e igual a: %d\n",area);
}
```

# Desafios

1. Elabore um programa, usando os comandos printf e scanf, que leia os valores do peso e da altura de uma pessoa, e ao final exiba na tela o valor do Índice de Massa Corporal (IMC) dessa pessoa. Observações:
   - Considere que o peso e a altura sejam dados em quilograma (kg) e metros (m), respectivamente.
   - IMC = peso / altura*altura;
   - Atente-se ao tipo de dado necessário das variáveis.
2. Escreva um programa em C que leia uma temperatura em graus celsius e imprima na tela a temperatura equivalente em farenheit. Mais uma vez, atente-se pelo tipo de dados necessário para o problema.
   Dado: F = 9C/5+32;
3. **Dica:** Pesquise sobre estruturas condicionais e sobre o comando return.
   A FCC (Fábrica de Cilindros de Carbono) fabrica vários tipos de cilindros de carbono. A FCC está instalada no décimo andar de um prédio, e utiliza os vários elevadores do préio para transportar os cilindros. Por questão de segurança, os cilindros devem ser transportados na posição vertical; como são pesados, no máximo dois cilindros podem ser transportados em uma única viagem de elevador. Os elevadores têm formato de

paralelepípedo e sempre têm altura maior que a altura dos cilindros.
Para minimizar o número de viagens de elevador para transportar os cilindros, a FCC quer, sempre que possível, colocar dois cilindros no elevador. A figura abaixo ilustra, esquematicamente (vista superior), um caso em que isto é possíbel (a), e um caso em que isto não é possível (b):




Como existe uma quantidade muito grande de elevadores e de tipos de cilindros, a FCC quer que você escreva um programa que, dadas as dimensões do elevador e dos dois cilindros, determine se é possível colocar os dois cilindros no elevador. **Entrada**
A entrada contém apenas um caso de teste. A primeira e única linha de cada caso de teste contém quatro números inteiros L, C, R1 e R2, separados por espaços em branco, indicando respectivamente a largura do elevador (1 <= L <= 100), o comprimento do elevador (1 <= C <= 100), e os raios dos cilindros (1 <= R1, R2 <= 100). Todos os dados de entrada são do tipo inteiro. Para o caso de teste, o seu programa deve imprimir uma única linha com um único caractere: 'S' se for possível colocar os dois cilindros no elevador e 'N' caso contrário. **Exemplo**

```
Entrada:        Saida:
11 9 2 3          S
7 8 3 2           N
10 15 3 7         N
8 9 3 2           S

```