# Declaração de variáveis

**Formato:** <tipo> <id1>, <id2>, ..., <idk>;
**O efeito da declaração:** reserva na memória RAM espaço para armazenar k variáveis do tipo <tipo>. Cada posição é identificada através dos identificadores de variáveis <id1>, <id2>, ..., <idk>. Os valores iniciais das variáveis definidas são indeterminados (lixo).

# Identificadores

São letras seguidas de números ou letras.
**Exemplos:** a, b, x1, x2, m, n, b12a4
**Observações:**

- Não podem haver dois identificadores iguais no main.
- Em C, as maiúsculas são diferentes de minúsculas. Podemos, por exemplo, ter as variáveis soma e sOma definidas num programa.

# Expressões

Envolvem os operadores + - * / % ( ).
**Exemplos**

- a + b*c
- 4*5%2 - 3 + c

Para saber em que ordem as contas são feitas, veja prioridade dos operadores.

# Prioridades dos operadores

As operações em C são feitas com as seguintes prioridades:

| | | | |
|---|---|---|---|
| ( ) | | | maior prioridade |
| * | / | % | |
| + | - | | menor prioridade |

Em expressões com operadores de mesma prioridade, as contas são feitas da esquerda para a direita.
**Exemplos**

- 5 + 4*3 é equivalente a 5 + (4*3) [óbvio!]
- 4*5%2 é equivalente a (4*5)%2
- 4*5/2*3%4/3 é equivalente a ((((4*5)/2)*3)%4)/3

# Comando de Atribuição

**Formato:** id = expressão;
**O que faz o comando**:

- o valor de expressão é calculado.
- o valor calculado é atribuído à variável id.

**Exemplos:**

- a = 5;
- b = 2*a + 3;
- a = b;

# Algumas abreviaturas em C

- ++i; é equivalente à atribuição i = i + 1;
- --i; é equivalente à atribuição i = i - 1;
- a += expressão; é equivalente à atribuição a = a + (expressão);. Ao invés de += também podemos usar -=, *=, /= e %=. Cuidado com a prioridade usada neste tipo de abreviatura. Por exemplo, a *= b + 5; é equivalente a a = a*(b + 5);.
- a = b = c = m = n = expressão; é equivalente a calcular o valor de expressão e depois atribuir esse valor a cada uma das variáveis a, b, c, m e n.
- int a, b = 0, c = 1, d = 5; é equivalente a declarar as variáveis inteiras a, b, c e d, inicializando os valores de b, c e d com os valores 0, 1 e 5, respectivamente. A variável a permanece com valor indeterminado.

"Apenas comentar partes deste tópico, revisar o mesmo ao utilizar em exercícios com %" **Operadores / e %**

Estaremos descrevendo os operadores / e % quando usado com operandos do tipo inteiro.
Se algum operando é de tipo ponto flutuante, o operador / tem o significado esperado e o operador % não pode ser usado.
O operador / calcula a parte inteira da divisão de seu primeiro operando (dividendo) pelo segundo (divisor).
Exemplos:

- O valor de 8/2 é 4.
- O valor de 7/2 é 3.
- O valor de 9/10 é 0.
- O valor de 19/10 é 1.
- O valor de 0/5 é 0.

O operador % calcula a parte inteira da divisão.
Exemplos:

- O valor de 7%2 é 1.
- O valor de 9%10 é 9.
- O valor de 20%10 é 0.
- O valor de 0%5 é 0.

**Observações:**

- Quando o denominador é 0 o resultado da operação é indefinido.
- Se os dois operandos são positivos, o resto da divisão é não-negativo e menor que o divisor.
- Se algum dos operandos é negativo, o resto da divisão tem valor absoluto menor que o valor absoluto do divisor, porém o resultado da operação varia de acordo com o processador usado. Por exemplo, -5/2 pode ser -2 ou -3 e -5%2 pode ser -1 ou +1.
- Sempre vale que (a/b)*b + a%b é igual a a.

# Condições

**Formato:** expressão1 comparador expressão2
**Como a condição é calculada**:

- os valores de expressão1 e expressão2 são calculadas
- esses valores são comparados de acordo com o comparador usado, sendo o valor da condição verdadeiro ou falso

Os comparadores que a gente vai usar: < > <= >= == !=
Note que o comparador da igualdade é o == e o operador de desigualdade é o !=.
**Exemplos de condições:**

- 1 > 0 (é sempre verdadeiro)
- 1 < 0 (é sempre falso)
- n + 3 != 5*m
- n <= 0

# Condições Compostas

As condições, simples ou compostas, podem ser usadas nos comandos while, if, if-else e for. Definimos em condições que uma condição tem o formato expressão1 comparador expressão2. Acrescentamos agora à essa definição que uma condição pode ser também de uma das seguintes formas:

- (condição1) && (condição2)
- (condição1) || (condição2)

**Como a condição é calculada**:

1. os valores de condição1 e condição2 são calculadas, tendo cada uma o valor verdadeiro ou falso
2. é feito o E ou o OU desses valores.

O valor resultante do passo (2) é calculado de acordo com a tabela abaixo.

| P | Q | Resultado Condição && | Resultado Condição \|\| |
|---|---|---|---|
| V | V | V | V |
| V | F | F | V |
| F | V | F | V |
| F | F | F | F |

**Exemplos de condições:**

- 1 > 0 (condição verdadeira)
- 1 < 0 (condição falsa)
- n + 3 != 5*m
- n <= 0
- n > 0 || n <= 0 (condição verdadeira)
- n > 5 || n <= 0 (condição verdadeira)
- n > 5 || m < 4 && k > 7

Nas condições compostas, muitas vezes os parênteses podem ser omitidos. As contas são feitas de acordo com as seguintes prioridades:

| | |
|---|---|
| ( ) | maior prioridade |
| * / % | |
| + - | |
| > < >= <= | |
| == != | |
| && | |
| \|\| | menor prioridade |

Portanto, a condição 1 > 0 || 3 < 2 && 5 < 4 é equivalente a (1 > 0) || ((3 < 2) && (5 < 4)) que é verdadeira. Já, a condição (1 > 0 || 3 < 2) && 5 < 4 é falsa.

# Comando printf

**Formato:** printf("mensagem", lista de expressões);
**O que faz o comando**: imprime na tela a mensagem, substituindo caracteres de controles.
A lista de expressões é uma lista de expressões separadas por vírgulas.
Os caracteres de controle são:

- **%d** tem a função de imprimir na tela o conteúdo da próxima expressão da lista de expressões. O valor é impresso como decimal. Mais tarde veremos que podemos usar também o **%f** e o **%c**.
- **\n** pula uma linha na tela. Ou seja, a próxima mensagem será mostrada na linha seguinte.

**Exemplos:**

- Imprime na tela Testando 3, 2, 1 e pula de linha.
- ```
  i = 1;
  printf("Testando %d, %d, %d\n", i+2, i+1, i);
  ```

- Imprime na tela soma = 84 e pula de linha. O valor de soma permanece inalterado, igual a 42.
- ```
  soma = 42;
  printf("soma = %d\n", 2*soma);
  ```

# Comando scanf

**Formato:** scanf("caracteres de controle", lista de variáveis);
**O que faz o comando**: interrompe o programa e espera que o usuário digite dados de entradas. Os dados de entrada devem ser terminados com a digitação da tecla enter. A cada carctere de controle deve corresponder uma variável a qual terá o valor que for digitado pelo usuário.
A lista de variáveis é uma lista de identificadores de variáveis separadas por vírgulas. Cada identificador deve ser precedido pelo caractere &, também conhecido como o "e comercial".
Os caracteres de controle são uma seqüência de **%d** . Mais tarde veremos que podemos usar também o **%f** e o **%c**.
**Exemplos:**

- Espera que o usuário digite um inteiro. O valor digitado será o conteúdo da variável n.
- ```
  scanf("%d", &n);
  ```

- Espera que o usuário digite dois inteiros. O primeiro valor digitado será o conteúdo da variável m e o segundo valor será o conteúdo da variável n.
- ```
  scanf("%d %d", &m, &n);
  ```

- O usuário deve digitar n números. Note que o printf tem como finalidade somente orientar o usuário para a digitação dos números.
- ```
  printf("Digite mais um número: ");
  scanf("%d", &num);
  ```
  ```

# Esqueleto de um Programa em C

```
//1º Versão
#include <stdio.h>
#include <stdlib.h>

int main()
{
 <declaração de variáveis>

 <comandos>

 system("pause");
 return 0;
}
```

```
//2º Versão
#include <stdio.h>
#include <stdlib.h>

 <declaração de constantes>

int main()
{
 <declaração de variáveis>

 <comandos>

 system("pause");
 return 0;
}
```

Algumas explicações:

- #include <stdio.h> tem como finalidade incluir as definições da biblioteca do C de entrada e saída: *STandarD Input Output*. Dessas funções, estaremos sempre usando o scanf e o printf.
- #include <stdio.h> tem como finalidade incluir as definições da biblioteca padrão do C: *STandarD LIBrary*. Dessas funções, estaremos sempre usando a função system. Essa função executa um comando do DOS, o comando pause. A finalidade é esperar para algo seja digitado antes da janela do DOS se fechar.
- int main() Como veremos mais tarde, um programa em C pode ter várias partes. Uma delas tem que se chamar main. É dessa parte que os comandos do programa começam a ser executados.
- return 0; mais tarde a gente fala sobre isso...

# Exercícios

1. Dados dois inteiros, determinar a sua soma.

**Solução**

```
#define <stdio.h>
#define <stdlib.h>

int main()
{
 int a, b, soma;

 /* Leitura dos dados */
 printf("Digite dois inteiros: ");
 scanf("%d %d", &a, &b);

 /* Faz a soma e armazena na variável soma */
 soma = a + b;

 /* Imprime a resposta */
 printf("A soma e': %d\n", soma);

 system("pause");
 return 0;
}
```

# Comando if

**Formato:**

```
if (condição)
  comando
```

**O que faz o comando**:

1. o valor de condição é calculado como verdadeiro ou falso
2. se a condição é verdadeira o comando é executado
3. se a condição é falsa, então o próximo comando no programa é executado.

**Exemplos:**

- Se a > b, então Maior é impresso.

```
if (a > b)
  printf("Maior\n");
```

- Se a > b, então Maior é impresso e é feita a atribuição a = 2*b.

```
if (a > b){
  printf("Maior\n");
  a = 2*b;
}
```

# Comando if-else

**Formato:**

```
if (condição)
  comando1
else
  comando2
```

**O que faz o comando**:

1. O valor de condição é calculado como verdadeiro ou falso.
2. Se a condição é verdadeira somente o comando1 é executado.
3. Se a condição é falsa somente o comando2 é executado.

**Exemplos:**

- Se a > b, então Maior é impresso; senão Menor é impresso.

```
if (a > b)
  printf("Maior\n");
else
  printf("Menor\n");
```

- Se a > b, então Maior é impresso e é feita a atribuição a = 2*b; se a < b, então Menor é impresso.

```
if (a > b){
  printf("Maior\n");
  a = 2*b;
}
else
  printf("Menor\n");
```

- Se a > b, então Maior é impresso; se a < b, então Menor é impresso e é feita a atribuição b = 3*a

```
if (a > b)
  printf("Maior\n");
else {
  printf("Menor\n");
  b = 3*a;
}
```

# Comando while

**Formato:**

```
while (condição)
  comando
```

**O que faz o comando**:

1. o valor de condição é calculado como verdadeiro ou falso
2. se a condição é verdadeira:
   i. o comando é executado
   ii. volte ao passo (1.)
3. se a condição é falsa, então o while é encerrado.

**Exemplos:**

- Imprime os números 0 1 2 3 4 um em cada linha:

```
i = 0;
while (i < 5){
  printf("%d\n",i);
  i = i + 1;
}
```

- Lê e imprime 10 números:

```
i = 0;
while (i < 10) {
  printf("Digite um número: ");
  scanf("%d", &num);
  printf("Número lido: %d\n",num);
  i = i + 1;
}
```

# Exercícios

1. Dada uma sequência de inteiros não-nulos, terminada por um zero, determinar a soma dos elementos da sequência.

**Solução**

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int num, soma;

  printf("Digite um numero: ");
  scanf("%d", &num);
  soma = 0;
  while (num != 0){
    soma = soma + num;
    printf("Digite um numero: ");
    scanf("%d", &num);
  }

  printf("A soma e' %d\n", soma);
  system("pause");
  return 0;
}
```

2. Dados n > 0 e uma sequência com n inteiros, determinar a soma dos inteiros positivos da sequência.

**Solução**

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int num, soma, n, cont;

  printf("Digite a quantidade de numeros: ");
  scanf("%d", &n);
  soma = 0;
  cont = 0;
  while (cont < n){
    printf("Digite um numero: ");
    scanf("%d", &num);
    cont = cont + 1;
    if (num > 0)
      soma = soma + num;
  }

  printf("A soma dos positivos e' %d\n", soma);

  system("pause");
  return 0;
}
```

3. Dados n > 0 e uma seqüência com n inteiros, determinar a soma dos inteiros positivos e dos inteiros negativos da seqüência.

**Solução**

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int num, n, somapos = 0, somaneg = 0, cont = 0;

  printf("Digite a quantidade de numeros: ");
  scanf("%d", &n);

  while (cont < n){
    printf("Digite um numero: ");
    scanf("%d", &num);
    cont = cont + 1;
    if (num > 0)
      somapos = somapos + num;
    else
      somaneg = somaneg + num;
  }

  printf("A soma dos positivos e' %d\n", somapos);
  printf("A soma dos negativos e' %d\n", somaneg);

  system("pause");
  return 0;
}
```

# Comando for

**Formato:**

```
for (atrib1; condição; atrib2)
  comando
```

**O que faz o comando**: é definido em função do comando while, como descrito a seguir.

```
atrib1;
while (condição){
  comando
  atrib2;
}
```

**Exemplos:**

- O usuário deve digitar n números. No final será apresentado a soma dos n números digitados.

```
int i, num = 0, soma = 0;

for (i = 0; i < n; i++){
  printf("Digite mais um número: ");
  scanf("%d", &num);
  soma = soma + num;
}

printf("A soma = %d", soma);
```

- Imprime os números 0 1 2 3 4 todos na mesma linha:

```
for (i = 0; i < 5; i=i+1)
  printf("%d ",i);
```

É equivalente aos comandos:

```
i = 0;
while (i < 5){
  printf("%d ",i);
  i = i + 1;
}
```

- Lê e imprime 10 números:

```
for (i = 0; i < 10; i++)
{
  printf("Digite um número: ");
  scanf("%d", &num);
  printf("Número lido: %d\n",num);
}
```

É equivalente aos comandos:

```
 i = 0;
    while (i < 10)
   {
       printf("Digite um número: ");
       scanf("%d", &num);
       printf("Número lido: %d\n", num);
       i++;
    }
```

# Exercícios

1. Dados n > 0 e as notas de n alunos, determinar quantos ficaram de recuperação (30 ≤ nota < 50).

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
  int n, nota, rec, i;

  printf("Digite a quantidade de alunos: ");
  scanf("%d", &n);
  rec = 0;
  for (i = 1; i <= n; i++) {
    printf("Digite uma nota: ");
    scanf("%d", &a);
    if (30 <= nota && nota < 50)
      rec = rec + 1;
  }

  printf("%d alunos ficaram de recuperacao.\n", rec);

  system("pause");
  return 0;
}
```

2. Escrever seu nome na tela 10 vezes. Um nome por linha.

```
#include <stdio.h>
int main()
{
    int x;

    for (x=1; x <=10; x++)
    printf("\nCurso de Redes\n");

    system("pause");
    return 0;
}
```

3. Receber do teclado um nome e imprimir tantas vezes quantos forem seus caracteres.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    int x, tam; char nome[30];

    printf("Digite um nome: ");
    gets(nome);

    tam = strlen(nome);
    for (x=1; x<=tam; x++)
    printf("\n%s\n\n",nome);
    system("pause");
    return 0;
}
```

# Declaração de Constantes Simbólicas

**Formato:** #define id valor
**O efeito da declaração**: as ocorrências de id no programa são substituídas por valor, com exceção das ocorrências que apareçam entre aspas ou façam parte de outros identificadores.
**Exemplo:**

```
#include <stdio.h>
#include <stdlib.h>

#define PI 3.1416
#define VERD 1    /* definicao nao usada no programa */
#define FALSO 0   /* definicao nao usada no programa */
#define enquanto while
#define se if
#define senao else
#define principal main
#define imprima printf
#define leia scanf
#define devolva return
#define sistema system

int principal()
{
  int a;

  imprima("Digite um inteiro: ");
  leia("%d", &a);

  enquanto (a > 0){
    se (a > PI)
      /* Note que o primeiro PI que esta' entre aspas nao sera' substituido */
      imprima("%d maior que PI = %d\n", a, PI);
    senao
      imprima("%d menor ou igual a PI = %d\n", a, PI);
    imprima("Digite um inteiro: ");
    leia("%d", &a);
  }

  sistema("pause");
  devolva 0;
}
```