BCC 201 - Introdução à Programação

Comandos de Entrada e Saída e de Controle de Fluxo

Guillermo Cámara-Chávez
UFOP

Lembrando a aula passada . . . I

- As variáveis:
  - são caracterizados por um **tipo** e um **nome**,
  - devem **começar** com **uma letra** (maiúscula ou minúscula) ou subscrito (_),
  - nunca pode começar com um número,
  - C é uma linguagem *case-sensitive*,

Lembrando a aula passada . . . II

  - utilizar como nome uma palavra que indique para que será usada esse variável.

- Constantes: são valores que não mudam durante a execução de um programa
- Para mostrar o conteúdo de uma variável podemos usar o comando `printf` (C)

A função *scanf* I

- realiza a leitura de um texto a partir do teclado
- parâmetros:
  - uma *string*, indicando os **tipos das variáveis** que serão lidas e o formato dessa leitura.
  - uma **lista de variáveis**
- aguarda que o usuário digite um valor e atribui o valor digitado à variável

A função *scanf* II

```c
#include <stdio.h>
int main(){
    int n;
    printf("Digite um número: ");
    scanf("%d", &n);
    printf("O valor digitado foi %d \n", n);
    return 0;
}
```

A função *scanf* III

O programa acima é composto de quatro passos:

- Cria uma variável n;
- Escreve na tela "Digite um número:"
- Lê o valor do número digitado
- Imprime o valor do número digitado

A função *scanf* IV

Memória

Endereço	Conteúdo
1000	
1001	
1002	
1003	
1004	
1005	
1006	
1007	
1008	
1009	
1010	
1011	
1012	
1013	
1014	
1015	
1016	
1017	
1018	

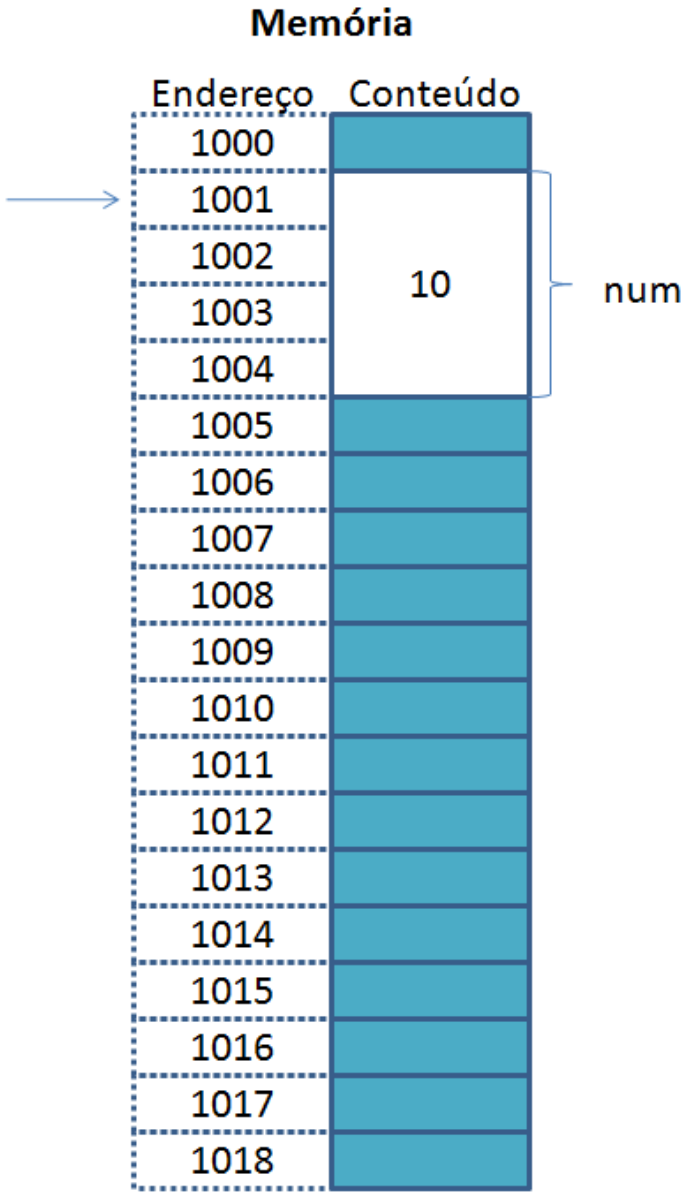
Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

Tela

Ação

A função *scanf* V



Código

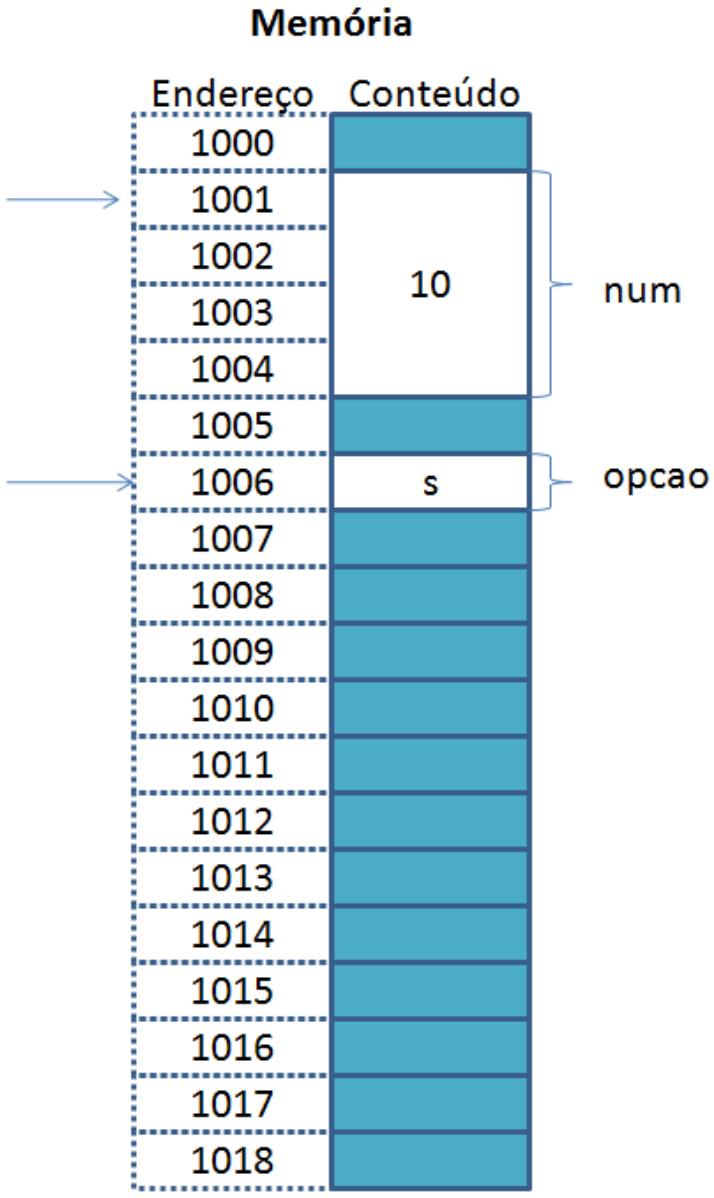
```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

Tela

Ação

Declara e separa memória para a variável ***num*** do tipo inteiro
Atribui o valor 10

A função *scanf* VI



Código

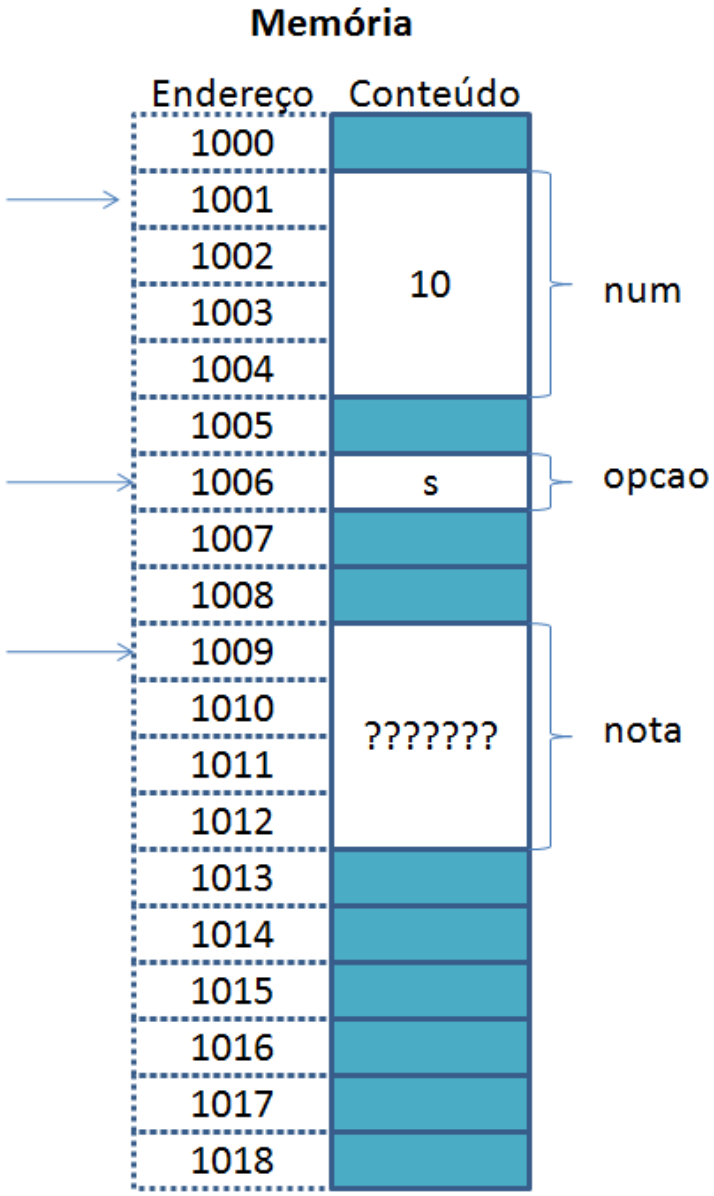
```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

Tela

Ação

Declara e separa memória para a variável ***opcao*** do tipo caracter
Atribui o valor 's'

A função *scanf* VII



Código

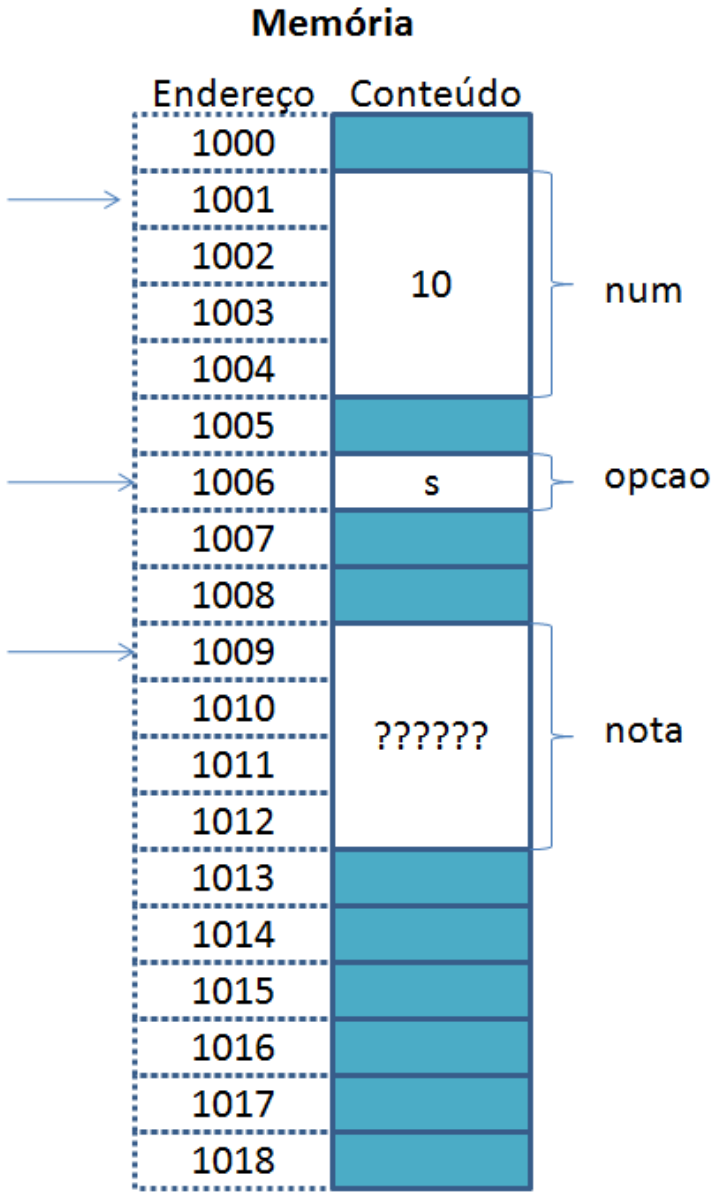
```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

Tela

Ação

Declara e separa memória para a variável ***nota*** do tipo real de precisão simples

A função *scanf* VIII



Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

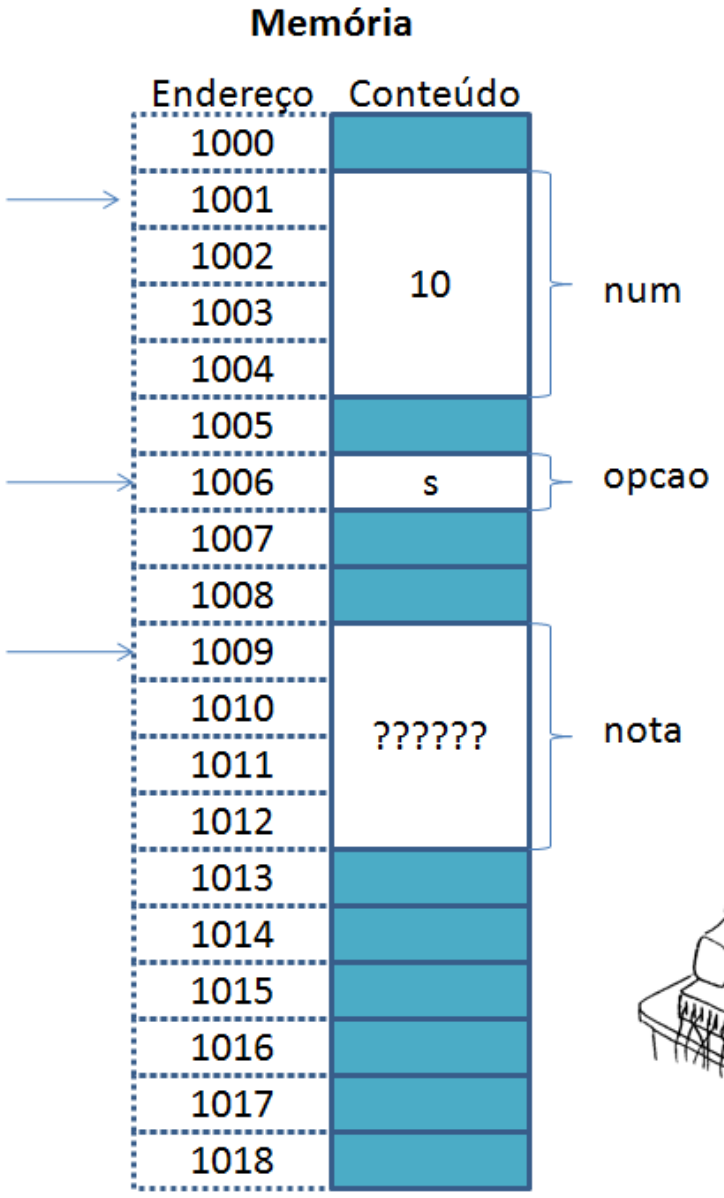
Tela

Inserir nota

Ação

Mostra na tela a mensagem : "Inserir nota"

A função *scanf* IX



Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

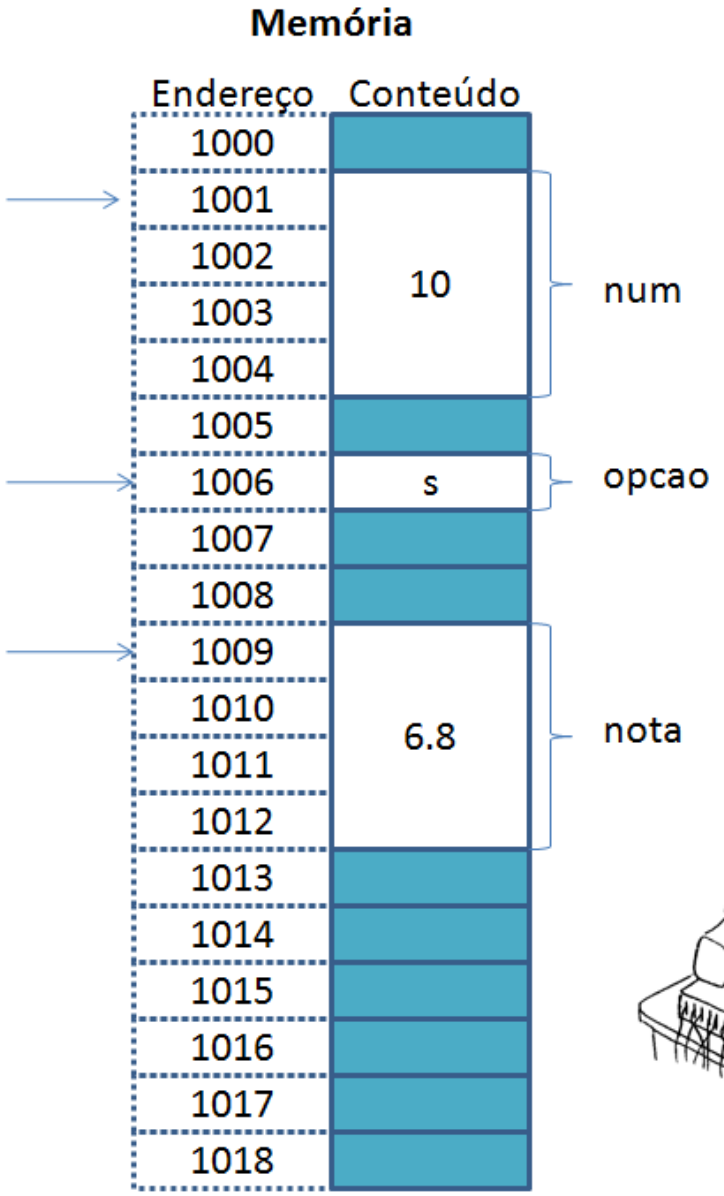
Tela

Inserir nota _

Ação

Operação de leitura, um valor número de tipo real será inserido no endereço de memória **1009**

A função *scanf* X



Código

```
int num = 10;  // 4 Bytes
char opcao = 's';  // 1 Byte
float nota;  // 4 Bytes
printf("Inserir nota");
scanf("%f", &nota);
```

Tela

Inserir nota 6.8

Ação

Usuário digita um número

A função *scanf* XI

Leitura de várias variáveis

```
#include <stdio.h>
int main(){
    int m, n, o;
    printf("Digite três números: ");
    scanf("%d %d %d",&m, &n, &o);
    printf("O valores digitados foram %d %d %d\n",
            m, n, o);
    return 0;
}
```

O endereço de uma variável I

- Toda variável tem um **endereço de memória** associada a ela.
- Esse endereço é o **local** onde essa variável é armanenada no sistema (como se fosse o endereço de uma casa, o local onde as pessoas são "armazenadas").

Endereço	Conteúdo	Variável
⋮	⋮	⋮
1001	???	
1002	2450	i
1003	???	
1004	225.345	f
1005	11331	j

O endereço de uma variável II

- Normalmente, o endereço das variáveis não são conhecidos quando o programa é escrito.
- O endereço é dependente do sistema computacional e também da implementação do compilador C que está sendo usado.
- O endereço de uma mesma variável pode mudar entre diferentes execuções de um mesmo programa C usando uma mesma máquina.

O operador “address-of” & de C I

- O operador & retorna o endereço de uma determinada variável

```
printf("%x", &valor);
```

- Imprime o endereço da variável `valor`.

O operador “address-of” & de C II

- É necessário usar o operador & no comando `scanf`, pois esse operador indica que o valor digitado deve ser colocado no endereço referente a uma variável.
- Esquecer de colocar o & comercial é um erro muito comum que pode ocasionar erros de execução.

O operador “address-of” & de C III

O programa abaixo imprime o valor e o endereço da variável:

```
#include <stdio.h>
int main(){
    int n = 8;
    printf("valor %d, endereço %x \n", n, &n);
    return 0;
}
```

Formatos de leitura de variável I

Os formatos de leitura são muito semelhantes aos formatos de escrita utilizados pelo `printf`. A tabela a seguir mostra alguns formatos possíveis de leitura.

Código	Função
%*c*	Lê um único caracter
%*s*	Lê uma série de caracteres (*string*)

Formatos de leitura de variável II

Código	Função
$\%d$	Lê um número inteiro
$\%u$	Lê um número inteiro positivo
$\%l$	Lê um inteiro longo
$\%f$	Lê um número real de precisão simples
$\%lf$	Lê um número real de precisão dupla

Formatos de leitura de variável III

Digite um caracter na tela

```
#include <stdio.h>
int main(){
    char car;
    printf("Digite um caracter: ");
    scanf("%c", &car);
    printf("O caracter digitado foi: %c \n", car);
    return 0;
}
```

Formatos de leitura de variável IV

Crie um programa que permita inserir o nome e a idade de um aluno

```
#include<stdio.h>
int main(){
    char nome[50];
    int idade;
    printf("\n Inserir nome: ");
    scanf("%s", nome);
    printf("\n Inserir idade: ");
    scanf("%d", &idade);
    printf("\n %s tem %d anos ", nome, idade);
    return 0;
}
```

Formatos de leitura de variável V

Crie um programa que permita inserir o nome completo (nome e sobrenome) de um aluno e a idade

```
#include<stdio.h>
int main(){
    char nome[50];
    int idade;
    printf("\n Inserir nome completo: ");
    scanf("%s", nome);
    printf("\n Inserir idade: ");
    scanf("%d", &idade);
    printf("\n %s tem %d anos ", nome, idade);
    return 0;
}
```

O programa funciona quando é digitado um espaço em branco?

Formatos de leitura de variável VI

A função `scanf` não aceita espaços em branco. A função `fgets` aceita todo tipo de caracter, incluindo o espaço em branco.
Sintaxe:

```
char * fgets ( char *str, int num, FILE* stream );
```

```
#include<stdio.h>
int main(){
    char nome[50];
    int idade;
    printf("\n Inserir nome completo: ");
    fgets(nome, 50, stdin);
    printf("\n Inserir idade: ");
    scanf("%d", &idade);
    printf("\n %s tem %d anos ", %s, %d);
    return 0;
}
```

Formatos de leitura de variável VII

- O comando `fgets` também permite inserir espaços em branco
  Sintaxe: `char* gets ( char *str );`
- Não é recomendável o uso da função `gets`
- Não existe forma de controlar a quantidade de caracteres a serem inseridos
- Utilizar uma função mais segura como `fgets`

Atribuição I

Atribuir um valor de uma expressão a uma variável **significa calcular o valor** daquela expressão e **copiar** aquele valor para uma **determinada variável**.

Atribuição II

No exemplo abaixo, a variável soma recebe o valor calculado da expressão `a + b`

```
soma = a + b;
```

Atribuição III

O operador de atribuição é o sinal de igual (=)

A esquerda do operador de atribuiçãao deve existir omente o nome de uma variável.

=

A direita, deve haver uma expressão cujo valor será calculado e armazenado na variável.

Expressão I

- Uma expressão é um conjunto de operações aritiméticas, lógicas ou relacionais utilizados para fazer "cálculos" sobre os valores das variáveis.
  Ex: `a + b`
  Calcula a soma de *a* e *b*

Expressão II

- Uma constante é uma expressão e como tal, pode ser atribuída a uma variável (ou em qualquer outro lugar onde uma expressão seja necessária)
  Ex: `a = 10;`
- Uma variável também é uma expressão
  Ex: `a = b;`

Expressão III

- `<expressão> + <expressão>`: calcula a soma de duas expressões.
  Ex. `a = a + b;`
- `<expressão> - <expressão>`: calcula a subtração de duas expressões.
  Ex. `a = a - b;`
- `<expressão> * <expressão>`: calcula a multiplicação de duas expressões.
  Ex. `a = a * b;`

Expressão IV

- `<expressão> / <expressão>`: calcula o quociente de duas expressões.
  Ex. `a = a / b;`
- `<expressão> % <expressão>`: calcula o resto da divisão inteira de duas expressões.
  Ex. `a = a % b;`

Expressão V

Exercício:

1. Qual é o valor da expressão `5+10%3`?
2. Qual é o valor da expressão `5*10%3`?

Precêdencia I

- Precedência é a ordem na qual os operadores serão calculados quando o programa for executado. Em C, os operadores são calculados na seguinte ordem:
  - `*` e `/`, na ordem em que aparecem na expressão.
  - `%`
  - `+` e `-`, na ordem em que aparecem na expressão.

Alterando a precedência I

- (<expressão>) também é uma expressão, que calcula o resultado da expressão dentra dela para só então permitir que as outras expressões executem.
- Deve ser utilizada quando a ordem da precedência não atende aos requisitos do programa.
  Ex. `5 + 10 % 3` retorna 6, enquanto `(5+10) % 3` retorna 0.
- Você pode usar quantos parênteses desejar dentro de uma expressão, contanto que utilize o mesmo número de parênteses para abrir e fechar expressões.

Operadores I

- Aritméticos: `+ - * /` e o sinal negativo: `-` ;
- Relacionais:

Operador	**Significado**	**Operador**	**Significado**
&&	e (*and*)	||	ou (*or*)
!	não (*not*)	<	menor
<=	menor igual	>	maior
>=	maior igual	==	igual
! =	diferente		

- Operadores de Ponteiros: * (acessa o conteúdo do ponteiro), & (obtém o endereço de uma variável);

Operadores II

- Incrementais: ++ (incrementa) ; -- (decrementa);
  - ++a: incrementa primeiro a e depois a usa (pré-fixado)
  - a++: primeiro usa a variável e depois a incrementa (pós-fixado)
- Atribuição, combinação dos outros operadores, são eles: $=$, $+=$, $-=$, $*=$, $/=$, %=

Incremento(++) e Decremento(–) I

- Operadores de incremento e decremento tem duas funções: servem como uma expressão e incrementam ou decrementam o valor da variável ao qual estão associados em uma unidade. Ex: c++ incrementa o valor da variável c em uma unidade
- Dependendo da posição do operador de incremento e decremento, uma função é executada antes da outra.

Incremento(++) e Decremento(–) II

- operador a direita da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão. Ex:

```
#include <stdio.h>
int main () {
    int a = 10;
    printf ("%d", ++a);
    return 0;
}
```

Qual é valor de a?

Incremento(++) e Decremento(–) III

- operador a direita da variável: Primeiro a variável é incrementada, depois a expressão retorna o valor da expressão. Ex:

```
#include <stdio.h>
int main () {
    int a = 10;
    printf ("%d", ++a);
    return 0;
}
```

Imprime 11

Incremento(++) e Decremento(–) IV

- operador a direita da variável: Primeiro a expressão retorna o valor da variável, e depois a variável é incrementada. Ex:

```
#include <stdio.h>
int main (void) {
    int a = 10;
    printf ("%d", a++);
    return 0;
}
```

Incremento(++) e Decremento(–) V

- operador a direita da variável: Primeiro a expressão retorna o valor da variável, e depois a variável é incrementada. Ex:

```
#include <stdio.h>
int main (void) {
    int a = 10;
    printf ("%d", a++);
    return 0;
}
```

Imprime 10

Incremento(++) e Decremento(–) VI

- Em uma expressão, os operadores de incremento e decremento são sempre calculados primeiro (tem maior precedência)

```
#include <stdio.h>
    int main (void) {
    int a = 10;
    printf ("%d", a * ++a);
    return 0;
}
```

Qual é o valor de *a*?

Incremento(++) e Decremento(–) VII

- Em uma expressão, os operadores de incremento e decremento são sempre calculados primeiro (tem maior precedência)

```
#include <stdio.h>
    int main (void) {
    int a = 10;
    printf ("%d", a * ++a);
    return 0;
}
```

Imprime 121

Incremento(++) e Decremento(–) VIII

Quais são os valores impressos?

```
#include <stdio.h>
int main () {
   int a = 10;
   printf("%d\n", a++);
   printf("%d\n", a);
   return 0;
}
```

Atribuições simplificadas I

Uma expressão da forma

```
a = a + b;
```

onde ocorre uma atribuição a uma das variáveis da expressão pode ser simplificada como

```
a += b;
```

Atribuições simplificadas II

Comando	Exemplo	Corresponde a:
$+=$	$a+=b$	$a=a+b;$
$-=$	$a-=b$	$a=a-b;$
$*=$	$a*=b$	$a=a*b;$
$\%=$	$a\%=b$	$a=a\%b;$

Atribuições simplificadas III

Qual é o valor mostrado na tela?

```
/*Operadores*/
int main(){
    int i = 10, j = 20;
    i = i + 1;
    i++;
    j -= 5;
    prinft("i + j = %d", i+j);
    return 0;
}
```

FIM