Pontifícia Universidade Católica
do Rio de Janeiro

Introdução a Programação de Jogos

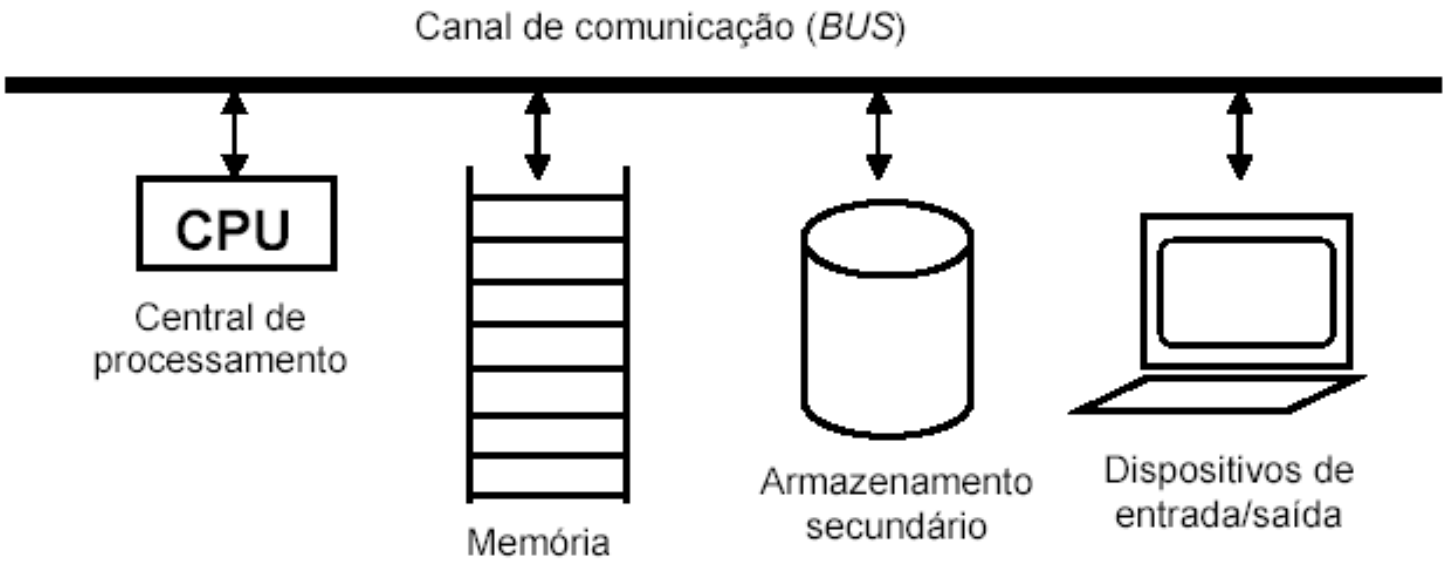
IUE1503

Aula 01 – Algoritmos e Ciclo de Desenvolvimento

Prof. Augusto Baffa

< abaffa@inf.puc-rio.br>

Modelo de um Computador



Linguagem de Máquina

- Um processador executa **instruções de máquina**
- Instruções de máquina são muito simples, mas são executadas em altíssima velocidade
- Um conjunto típico de instruções inclui operações aritméticas, lógicas e de desvio

- Exemplo de seqüência típica de instruções:
  1. `Carregue o conteúdo da posição de memória 40 no registrador 1 (R1);`
  2. `Carregue o valor 100 no registrador 2 (R2);`
  3. `Se o conteúdo de R1 for maior do que o conteúdo de R2 prossiga com a instrução armazenada na posição de memória 240;`

Linguagem de Máquina

- Instruções de máquina são representadas por seqüências de **dígitos binários**

- Exemplo:

Adicionar R1 e R2 e armazenar o resultado em R6

```
[  op  |  rs |  rt |  rd |shamt| funct]
   0      1     2     6     0     32      decimal
 000000 00001 00010 00110 00000 100000    binary
```

Desviar para a instrução armazenada no endereço de memória 1024

```
[  op  |         target address         ]
   2                  1024                decimal
 000010 00000 00000 00000 10000 000000    binary
```

Linguagens de Alto-Nível

- **Programar em linguagem máquina** é uma tarefa entediante e propensa a erros
- A partir de meados dos anos 50 várias **linguagens de alto nível** foram criadas
- Possuem **nível de abstração** relativamente elevados
- Elas são mais próximas das linguagens utilizadas pelos seres humanos

Linguagens de Alto-Nível

- **Exemplos de linguagens de alto-nivel:**
  - FORTRAN (1957)
  - COBOL (1960)
  - PASCAL (1970)
  - C (1972)
  - C++ (1983)
  - JAVA (1995)

Exemplo de Programa em C

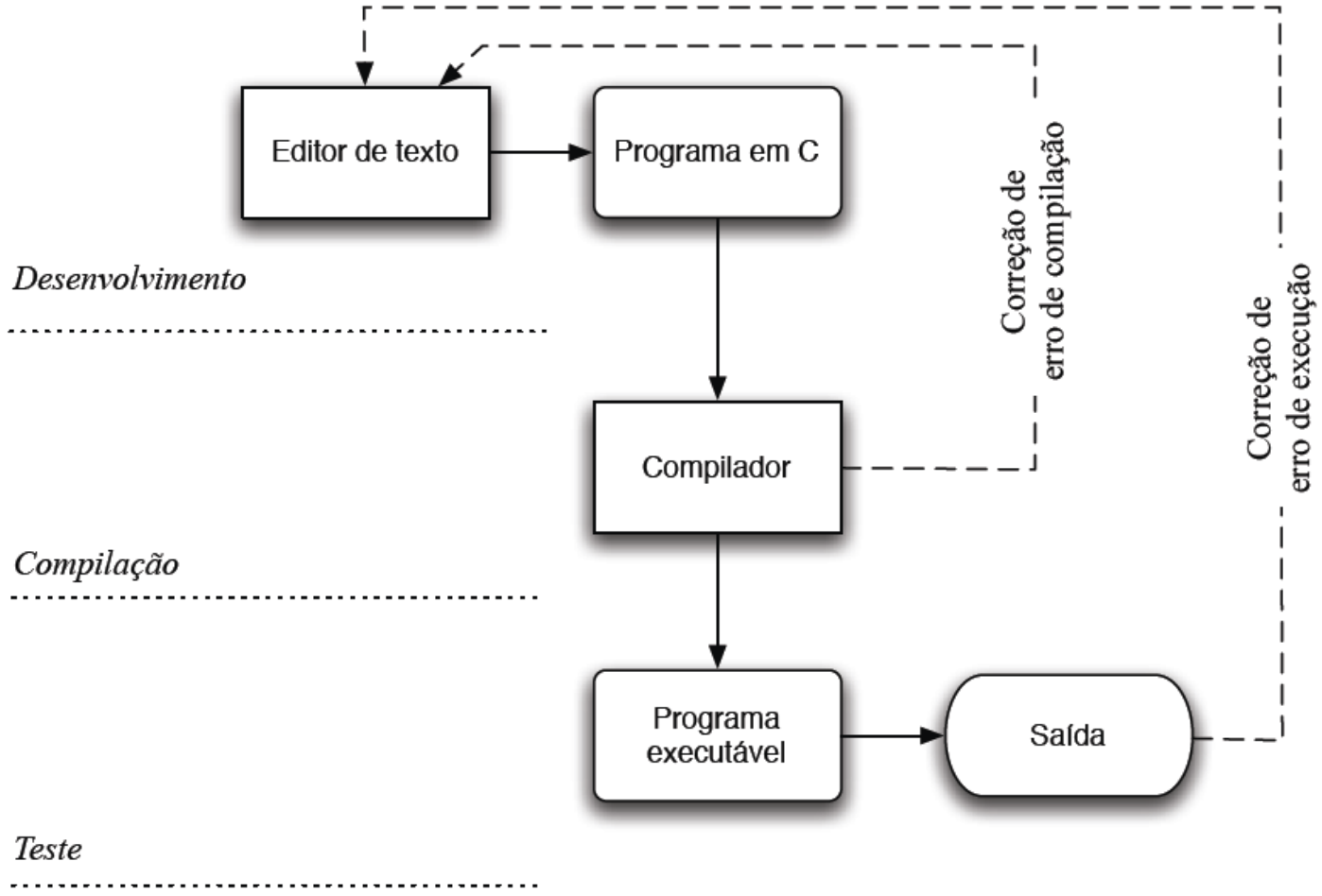
```
#include <stdio.h>

int main(void)
{
  int numero1, numero2, resultado;
  printf("Digite o primeiro numero: ");
  scanf("%d", &numero1);
  printf("Digite o segundo numero: ");
  scanf("%d", &numero2);
  resultado = numero1 + numero2;
  printf("Resultado da soma é %d", resultado);
  return 0;
}
```

Compilação

- Programas escritos em linguagens de alto nível **não são executados** diretamente pelo processador
- Antes de executá-lo é preciso **traduzir os comandos** de alto-nível para instruções de máquina
- Esta tarefa (**compilação**) é realizada por um programa chamado de **compilador**.
- Linguagem Compilada X Linguagem Interpretada

Ciclo de Desenvolvimento



Algoritmos

- **Lógica de Programação** é a técnica de criar sequências lógicas de ações para atingir determinados objetivos.
- **Sequências Lógicas** são instruções executadas para atingir um objetivo ou solução de um problema.
- **Instruções** são uma forma de indicar ao computador uma ação elementar a executar.
- Um **Algoritmo** é formalmente uma sequência finita de instruções que levam a execução de uma tarefa.

Algoritmos

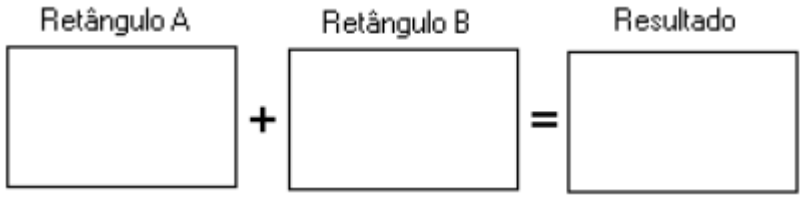
- Até mesmo tarefas simples, podem ser descritas por **sequências lógicas**:

"Chupar uma bala"

1) Pegar a bala;
2) Retirar o papel;
3) Chupar a bala;
4) Jogar o papel no lixo;

"Somar dois números"

1) Escreva o primeiro número no retângulo A.
2) Escreva o segundo número no retângulo B.
3) Some o número do retângulo A com número do retângulo B e coloque o resultado no retângulo C.



Escrevendo Algoritmos

- Os algoritmos podem ser escritos diretamente em uma linguagem de programação ou simplesmente descritos em pseudocódigo.

- **Pseudocódigo** é uma forma genérica de escrever um algoritmo.

- **Linguagens de programação** são formas padronizadas de comunicar instruções para o computador. São conjuntos de regras sintáticas e semânticas usadas para definir um programa de computador.

Escrevendo Algoritmos

Processo Geral de um Algoritmo

- **Entrada**: O algoritmo recebe os dados de entrada.
- **Processamento**: Os procedimentos para se chegar ao resultado final são executados.
- **Saída**: O resultado final é mostrado.

Escrevendo Algoritmos

- **Exemplo:** “Ler duas notas e calcular a média”

Algoritmo descritivo:

Leia a primeira nota e armazene ela em **nota1**;
Leia a segunda nota e armazene ela em **nota2**;
Some a **nota1** com a **nota2** e divida o **resultado** por 2;
Mostre o **resultado**;

Pseudocódigo:

```
variáveis
    nota1, nota2, media : real;

início
    escreva(“Digite a nota 1”);
    leia(nota1);
    escreva(“Digite a nota 2”);
    leia(nota2);
    media = (nota1 + nota2)/2;
    escreva(media);
fim
```

visionlab
visualization laboratory

Pontifícia Universidade Católica
do Rio de Janeiro

Introdução a Programação de Jogos

IUE1503

Aula 01 – Introdução a Linguagem C

Prof. Augusto Baffa

< abaffa@inf.puc-rio.br>

Estrutura de um Programa C

Inclusão de bibliotecas auxiliares: **#include <nome>**

Definição de constantes: **#define nome valor**

Funções auxiliares

Função Principal (início da execução de um programa): **int main(void)**

Bibliotecas Auxiliares

- **stdio.h**: funções de entrada de saída de dados:
  - printf, scanf...

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

- **math.h**: funções matemáticas:
  - cos, sen, sqrt, pow...

```
#include <math.h>
```

- **string.h**: funções de manipulação de texto (string):
  - strcmp, strlen...

```
#include <string.h>
```

Definição de Constantes

- A diretiva **#define** associa um identificador a um valor.
- **Formato:**

```
#define nome_macro valor
```

- É usual definir o nome das macros com letras maiúsculas
- Exemplos:

```
#define UM 1
#define MSG "Digite S para sair"
#define PI 3.1415
```

Funções Auxiliares

- As funções têm a seguinte estrutura:

Um programa C não pode ter duas funções com o mesmo nome.

```
tipo_de_retorno nome_da_funcao (parametros)
{
   variaveis locais

   instrucoes em C (comandos = expressoes e
   operadores)
}
```

Se uma função não tem retorno colocamos *void*.

Se uma função não tem uma lista de parâmetros colocamos *void* ou apenas o ().

Consiste no bloco de comandos que compõem a função.

Estrutura de um Programa C

- **Inclusão de bibliotecas auxiliares:**

```
#include <nome.h>
```

- **Função Principal:**

```
int main(void)
{
   ...
}
```

Função Principal

```
int main(void)
{
      /* declarações de variáveis locais,
      chamadas a funções auxiliares,
      cálculos de expressões, leitura e
      escrita de dados, etc. */
}
```

Observação: /* delimita um comentário em C */

Variáveis e Constantes

- **Variáveis** e **constantes** são os elementos básicos manipulados por um programa.

- **Constante** é um valor fixo que não se modifica ao longo da execução de um programa.

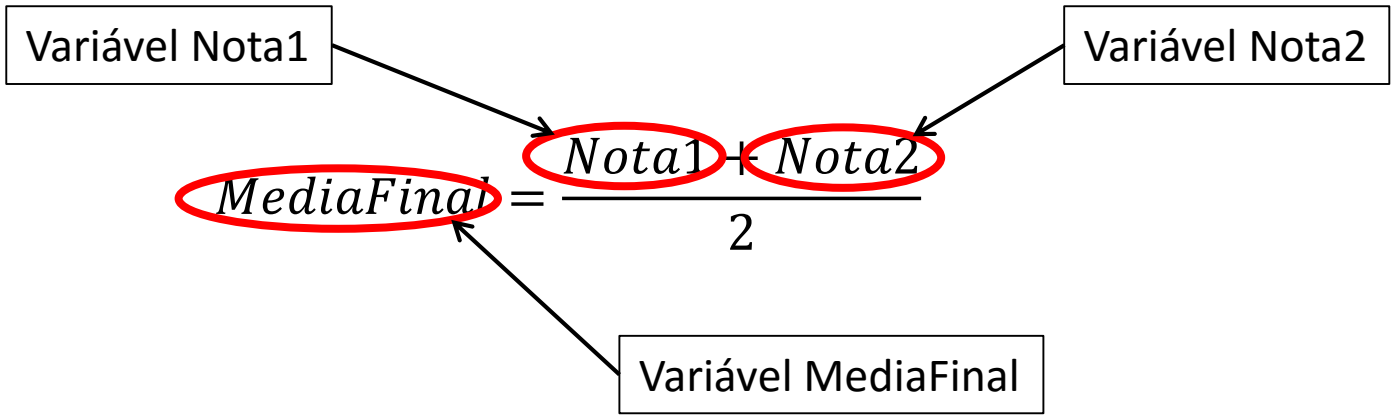
$$MediaFinal = \frac{Nota1 + Nota2}{2}$$

valor constante

Variáveis

- **Variável** é um espaço reservado na memória do computador para armazenar um determinado tipo de dado.

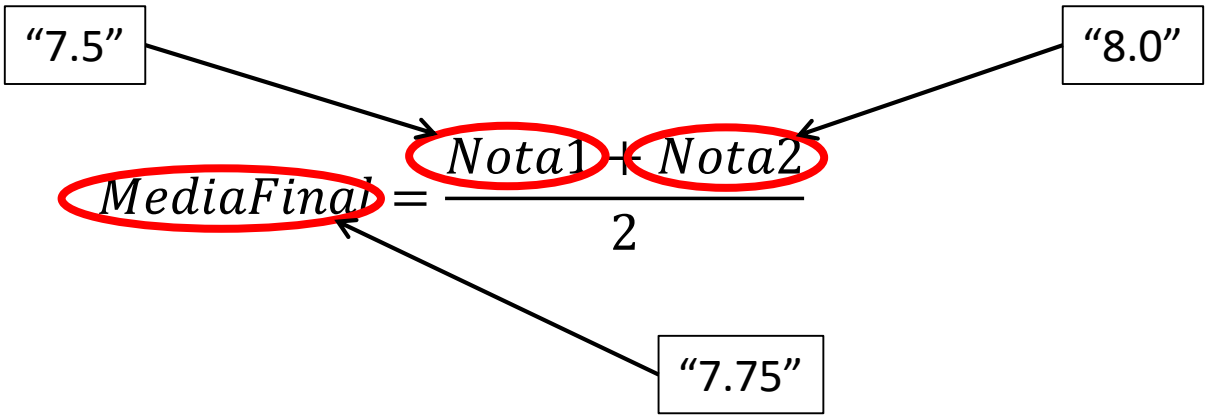
- Variáveis recebem **nomes** para serem referenciadas e modificadas quando necessário.



Variáveis

- O **conteúdo de uma variável** pode se modificado ao longo da a execução do programa.

- Embora uma variável possa assumir diferentes valores, ela só pode armazenar **um valor a cada instante**.



Variáveis em C

- **Variável** é um espaço reservado na memória do computador para armazenar um tipo de dado.
- Devem receber **nomes** para poderem ser referenciadas e modificadas quando necessário.
- Toda variável tem:
  - um nome
  - um tipo de dado
  - um valor
- **Restrição para nomes:** não é permitido começar o nome com um algarismo (0-9), alguns caracteres não são válidos (*, -, /, +, ...), e palavras reservadas não podem ser utilizadas (main, if, while, ...).

Variáveis

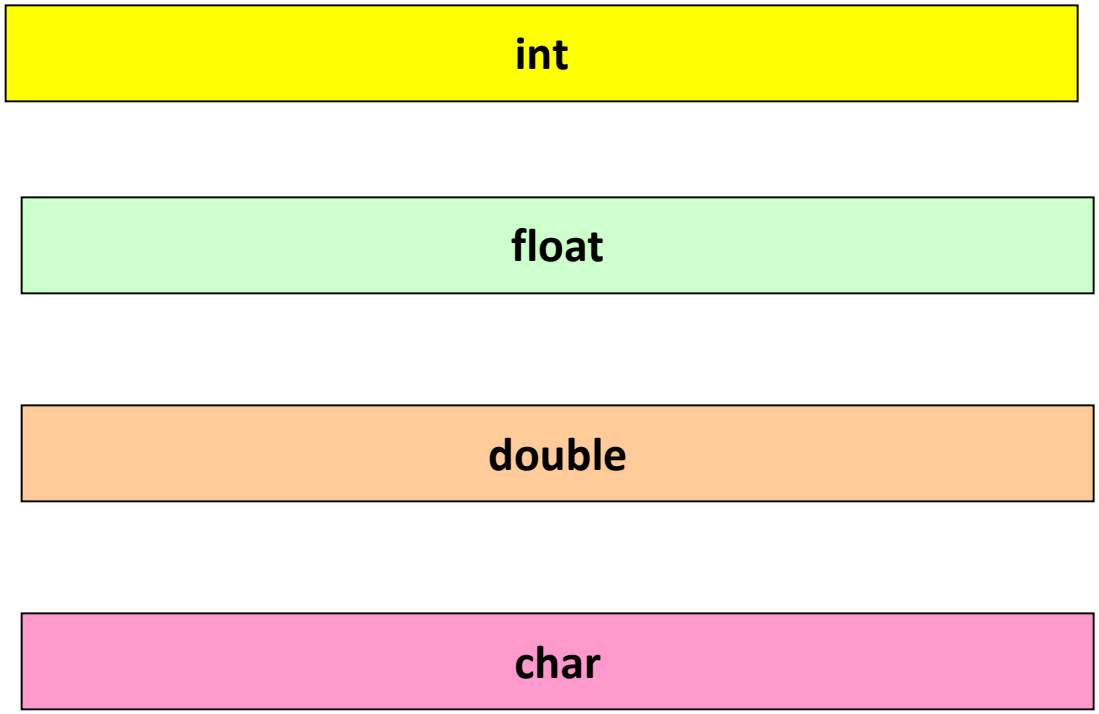
- É necessário informar o nome e o tipo das variáveis:
  - O compilador precisa saber o tipo para **alocar** (reservar) o espaço de memória pré-definido para aquele tipo (quantidade de *bytes*).
  - O nome será usado para representar o espaço que está sendo alocado.
- Exemplo:

```
int main (void)
{
   float nota1, resultado;
   …
}
```

Tipos de Variáveis da Linguagem C

Tipo	Tamanho	Menor valor	Maior valor
`char`	1 byte	-128	+127
`unsigned char`	1 byte	0	+255
`short int (short)`	2 bytes	-32.768	+32.767
`unsigned short int`	2 bytes	0	+65.535
`int (*)`	4 bytes	-2.147.483.648	+2.147.483.647
`long int (long)`	4 bytes	-2.147.483.648	+2.147.483.647
`unsigned long int`	4 bytes	0	+4.294.967.295
`float`	4 bytes	-10^{38}	$+10^{38}$
`double`	8 bytes	-10^{308}	$+10^{308}$

Principais Tipos de Variáveis



Declaração de Variáveis

- Variáveis devem ser explicitamente declaradas.
- Variáveis podem ser declaradas em conjunto.

Exemplos:

```
int a;      /* declara uma variável do tipo int */
int b;      /* declara uma variável do tipo int */
float c;    /* declara uma variável do tipo float */
int d, e;   /* declara duas variáveis do tipo int */
int d = 5; /* declaração e inicialização da variável */
```

Operadores Aritméticos

- **Operadores aritméticos** são usados para se realizar operações aritméticas com as variáveis e constantes.

Operação	Símbolo
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da Divisão	%

Exemplos:

operador de atribuição

```
total = preco * quantidade;
media = (nota2 + nota2)/2;
resultado = 3 * (1 - 2) + 4 * 2;
resto = 4 % 2;
```

operado apenas entre inteiros

Funções de Entrada e Saída em C

- **Função "printf":** Permite a saída de dados, ou seja, a escrita de dados na tela.

```
printf(formato, lista de constantes/variáveis/expressões...);
```

```
printf("%d %g", 33, 5.3);

tem como resultado a impressão da linha:
33 5.3
```

```
printf("Inteiro = %d   Real = %g", 33, 5.3);

com saída:
Inteiro = 33   Real = 5.3
```

Funções de Entrada e Saída em C

- Especificação de formatos:

Formato	Descrição
`%c`	Especifica um char
`%d`	**Especifica um int**
`%f`	**Especifica um float**
`%e`	Especifica um double (ou float) no formato científico
`%g`	Especifica um double (ou float) no formato mais apropriado (%f ou %e)
`%s`	Especifica uma cadeia de caracteres

Funções de Entrada e Saída em C

- Impressão de texto:

```
printf("Curso de Programação\n de Jogos");

exibe na tela a mensagem:
Curso de Programação
de Jogos
```

Funções de Entrada e Saída em C

- **Função "scanf":** Permite a entrada de dados, ou seja, a captura de valores fornecidos via teclado.

```
scanf(formato, lista de endereços das variáveis...);
```

```
int n;
scanf("%d", &n);

valor inteiro digitado pelo usuário é armazenado na
variável n
```

Funções de Entrada e Saída em C

- **Função “scanf”:**
  - caracteres diferentes dos especificadores no formato servem para separar a entrada
  - espaço em branco dentro do formato faz com que sejam "pulados" eventuais brancos da entrada
  - %d, %f, %e e %g automaticamente pulam os brancos que precederem os valores numéricos a serem capturados

```
scanf ("%d %d", &h, &m);

valores (inteiros) fornecidos em uma mesma linha devem
ser separados por espaço
```

Exemplo 01

- Escreva um programa que leia dois números inteiros e retorne a soma deles.

```
#include <stdio.h>

int main(void)
{
  int numero1, numero2, resultado;
  printf("Digite o primeiro numero: ");
  scanf("%d", &numero1);
  printf("Digite o segundo numero: ");
  scanf("%d", &numero2);
  resultado = numero1 + numero2;
  printf("Resultado da soma é %d", resultado);
  return 0;
}
```

Exemplo 1 – Execução Passo-a-Passo

- Comando:

numero1	numero2	resultado	saída
????	????	????	

Exemplo 1 – Execução Passo-a-Passo

- Comando:

```
printf("Digite o primeiro numero: ");
```

numero1	numero2	resultado	saída
????	????	????	Digite o primeiro numero:

Exemplo 1 – Execução Passo-a-Passo

- Comando:

```
scanf ("%d", &numero1);
```

numero1	numero2	resultado	saída
????	????	????	Digite o primeiro numero:
15	????	????	

Exemplo 1 – Execução Passo-a-Passo

- Comando:

```
printf("Digite o segundo numero: ");
```

numero1	numero2	resultado	saída
????	????	????	Digite o primeiro numero:
15	????	????	Digite o segundo numero:

Exemplo 1 – Execução Passo-a-Passo

- Comando:

```
scanf ("%d", &numero2);
```

numero1	numero2	resultado	saída
????	????	????	Digite o primeiro numero:
15	????	????	Digite o segundo numero:
15	3	????	

Exemplo 1 – Execução Passo-a-Passo

- Comando:

```
resultado = numero1 + numero2;
```

numero1	numero2	resultado	saída
????	????	????	Digite o primeiro numero:
15	????	????	Digite o segundo numero:
15	3	18	

Exemplo 1 – Execução Passo-a-Passo

- Comando:

```
printf("Resultado da soma é %d", resultado);
```

numero1	numero2	resultado	saída
????	????	????	Digite o primeiro numero:
15	????	????	Digite o segundo numero:
15	3	18	Resultado da soma é 18

Programando em C - Exemplo

- **Indentação e Comentários:**

```
/* Programa para converter temperatura de Celsius em Fahrenheit */
#include <stdio.h>

int main (void)
{
     float cels; /* armazena temperatura em oC */
     float fahr; /* armazena temperatura em oF */

     /* captura valor fornecido via teclado */
     printf("Digite a temperatura em Celsius: ");
     scanf("%f", &cels);

     fahr = 1.8 * cels + 32; /* faz a conversão */

     /* exibe resultado na tela */
     printf("Temperatura em Fahrenheit: %f", fahr);
     return 0;
}
```

Aritmética Inteira

- Qual o erro na expressão abaixo?

```
int res = 5/3;
```

  - **O erro não está na expressão e sim no resultado.** Como 5 e 3 são valores inteiros, o resultado dessa divisão é um número inteiro e portanto seria 1, e não 1.666...

- Outro exemplo:

```
int a = 5;
int b = 2;
float c = a/b;
```

  - **5/2 = 2.** Como o resultado está sendo armazenado em um float, o valor dessa divisão em c é 2.0

Conversão de Tipo

- Se temos duas variáveis inteiras e queremos que a divisão seja feita em representação real, podemos usar um **operador de conversão de tipo**:

```
int a = 5;
int b = 2;
float c = (float) a/b;
```

  - Operações aritméticas são feita na representação do **tipo de maior expressividade**: double > float > int
  - O **(float)** converte **a** em 5.0 e depois divide por 2
  - Note que o valor da variável continua sendo do tipo inteiro e o valor de a não é alterado

Conversão de Tipo

- Outro casos em que a conversão de tipo também é útil:

```
int a;
float b = 2.6;
a = b;
```

O compilador vai gerar uma mensagem de "*warning*". Para evitar:

```
int a;
float b = 2.6;
a = (int) b;
```

Exercício

1) Crie um programa que faça a multiplicação dois números digitados pelo usuário

   a) Escreva o algoritmo de forma descritiva

   b) Escreva o algoritmo em pseudocódigo

Exercício

2) Crie um programa que calcule a área de um triângulo

$$Area = \frac{b * h}{2}$$

a) Escreva o algoritmo de forma descritiva

b) Escreva o algoritmo em pseudocódigo

Exercícios

Lista de Exercícios 01 - Variáveis

http://www.inf.puc-rio.br/~abaffa/iue1503/