UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Sistemas de Informação

Linguagem de Programação I

Professor : Carlos Helano

Objetivo

- Desenvolver a capacidade lógica para construção de algoritmos na resolução de problemas.
- Apresentar uma visão geral do processo de programação.
- Apresentar os recursos da linguagem de programação C.

Conteúdo

- Conceitos básicos da programação.
- Introdução a Linguagem C (Conceitos básicos)
  - Tipos de Dados;
  - Operações;
  - Comandos de Entrada e Saída de Dados;
- Estruturas de Controle
  - Estruturas sequenciais;
  - Estruturas condicionais;
  - Estruturas de repetição;
- Estrutura de Dados Compostas
  - Vetores, Matrizes, Registros (Struct)

Ferramentas para Prática

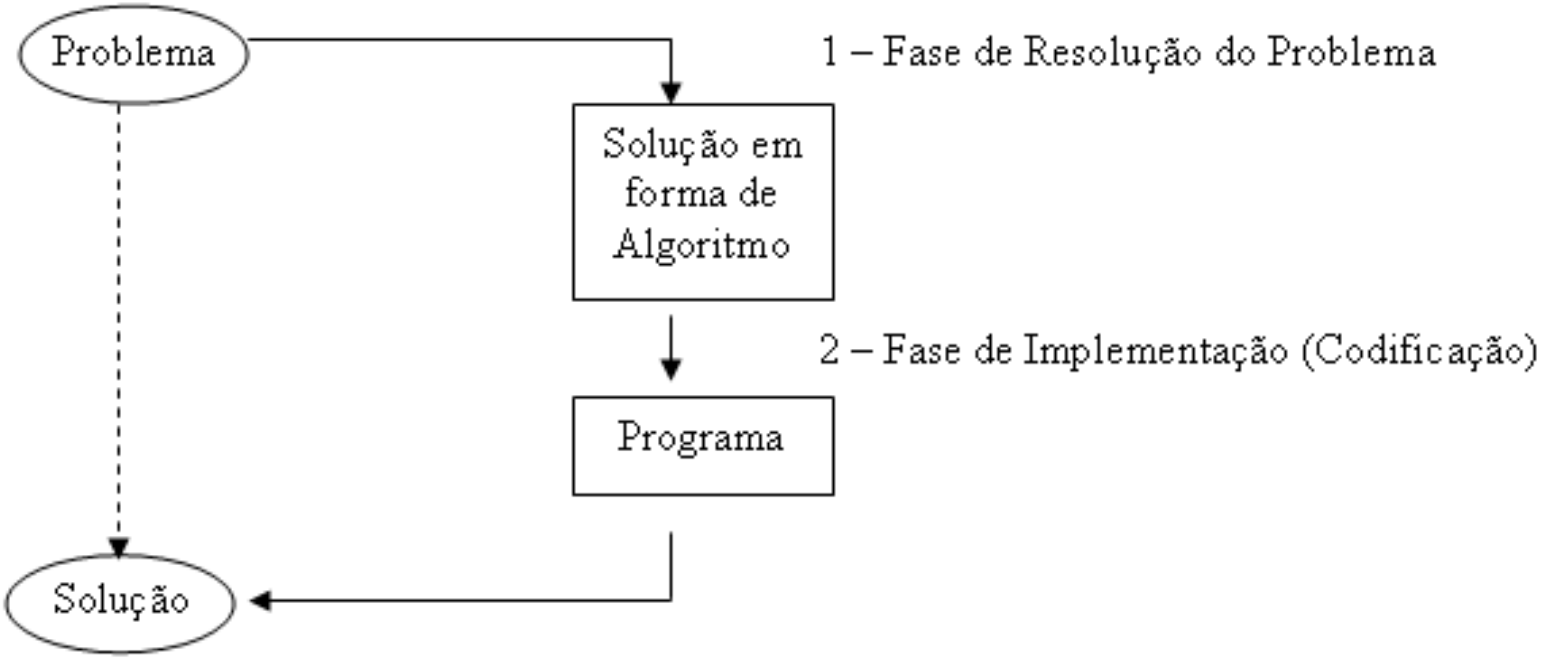
- Utilizaremos as ferramentas abaixo para o desenvolvimento das práticas
  - Falcon C++ . Disponível : https://sourceforge.net/projects/falconcpp/
  - Repl.it. Disponível : https://repl.it/

Conceitos Básicos

- Algoritmos
  - A ideia de algoritmos surge pela necessidade do homem em resolver problemas.
  - A tarefa de Processamento de Dados da máquina consiste em tomar certa informação, processá-la e obter o resultado desejado.

Conceitos Básicos

- Como utilizar o algoritmo para solução de problemas



Conceitos Básicos

Situação Problema :

– Calcular a média de 3 Provas. P1, P2, P3

- 1 – Identificar as Entradas

  P1, P2, P3
- 2 – Processamento

  (P1+P2+P3)/3
- 3 – Saída

  Média Final

Algoritmos estruturados

Estruturas de um Algoritmo (Estruturas de Controle)

- 1 – Sequencial
  - (Os Comandos serão executados um após o outro)
- 2 – Condicional
  - (Os Comandos só serão executados se uma determinada condição for satisfeita)
- 3 – Repetitiva
  - (Os Comandos serão executados enquanto uma determinada condição for satisfeita)

Conceitos Básicos

- Definição : Algoritmos Estruturados

*Conjunto de comandos que, obedecidas as estruturas de controle, resultam numa sucessão finita de ações. (**Visam um objetivo bem definido**)*

Algoritmos estruturados

Regras para um bom Algoritmo

- 1 – Ações simples e bem definidas
- 2 – Seqüência ordenada de ações
- 3 – Seqüência finita de ações

Algoritmos estruturados

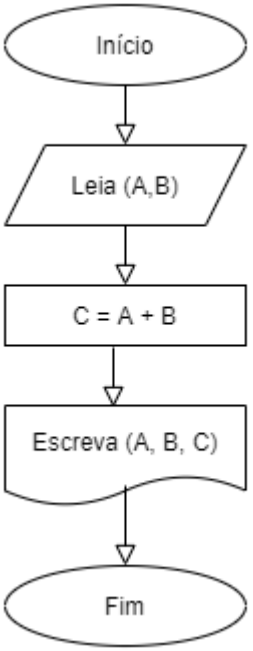
Fluxograma

- Permite a representação gráfica de um algoritmo
- Permite que pessoas diversas participem do desenvolvimento. Ou melhor, pessoas de diversas áreas do conhecimento podem contribuir na construção do algoritmo.

Estruturas de Controle

- 1 – Seqüencial
  - (Os Comandos executados um após o outro)

```
Inicio Algoritmo
    Leia (A)
    Leia (B)
    C = A + B
    Escrever (A,B,C)
Fim Algoritmo
```

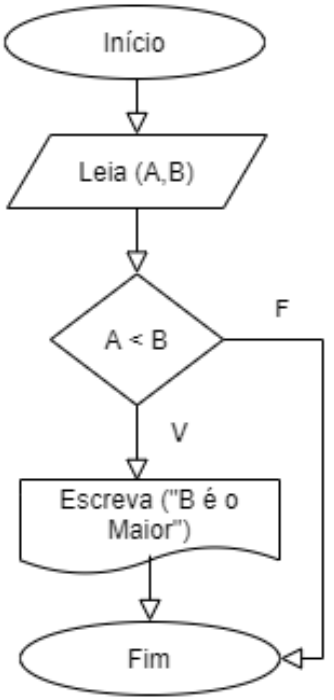


Estruturas de Controle

- 2 – Condicional
  - (Os Comandos só serão executados se uma determinada condição for satisfeita)

```
Se condição então
        Comandos
Fim Se
```

```
Inicio Algoritmo
        Leia (A,B)
        Se A < B então
                Escrever "B é o Maior"
        Fim se
Fim Algoritmo
```

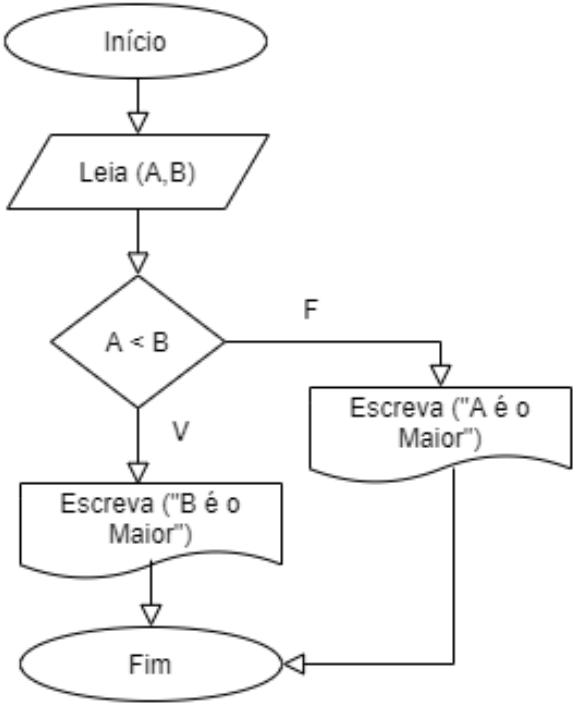


Estruturas de Controle

- 2 – Condicional
  - (Os Comandos só serão executados se uma determinada condição for satisfeita)

```
Se condição então
    Comandos
Senão
    Comandos
Fim Se
```

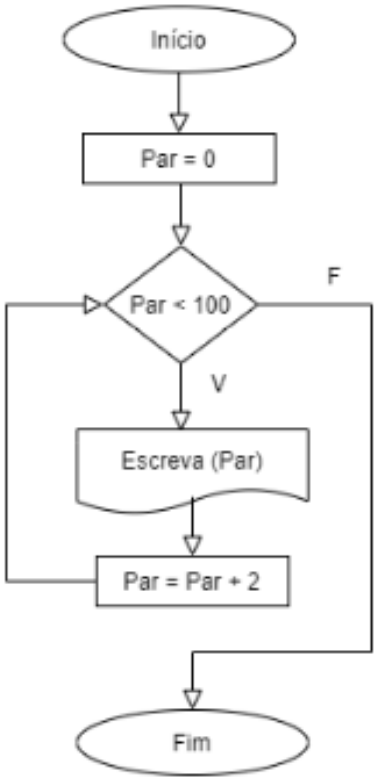
```
Inicio Algoritmo
    Leia (A,B)
    Se A < B então
        Escreva "B é o Maior"
    Senão
        Escreva "A é o Maior"
    Fim Se
Fim Algoritmo
```



Estruturas de Controle

- 1 – Repetição
  - (Os Comandos só serão executados se uma determinada condição for satisfeita)

```
Inicio Algoritmo
    Par = 0
    Enquanto Par < 100 faça
        Escrever Par
        Par = Par + 2
    Fim Enquanto
Fim Algoritmo
```



Exercício

Problema

01 - Faça um programa que receba o salário-base de um funcionário, calcule e mostre o salário a receber. Sabendo-se que esse funcionário tem gratificação de 5% sobre o salário-base e paga imposto de 7% também sobre o salário-base.

Linguagem de Programação C - Histórico

- Criada entre 1969 e 1973 por Dennis Ritchie.
  - A **linguagem C** foi uma evolução da **Linguagem B** Criada por Ken Thompson.
  - A linguagem foi desenvolvida para o Sistema Operacional UNIX.

Ken Thompson e Dennis Ritchie

- Em 1978 foi publicada a primeira edição do livro "*The C Programming Language*", por Brian Kernighan e Dennis Ritchie.
- Com a popularização dos microcomputadores muitas implementações foram feitas em C, sem um padrão definido.
- Em 1983, o ANSI (*American National Standards Institute*) estabeleceu um comitê para padronização da Linguagem C.

Linguagem de Programação C - Histórico

- Durante a década de 1980, Bjarne Stroustrup começou a trabalhar em um projeto onde se adicionavam construções de linguagens de programação orientada por objetos à linguagem C. Surgia a chamada Linguagem C++.

- Linguagens Influenciadas pela linguagem C :

Visão Geral – Linguagem C

- A Linguagem C é considerada uma linguagem de **Nível Médio**.
  - **As Linguagens de Baixo Nível** compreendem características da Arquitetura do Computador. (Linguagens de Primeira e segunda Geração).
  - **As Linguagens de Alto Nível** são linguagens com um nível de abstração relativamente alto. Elas estão mais próximas da linguagem humana.

Nível Alto	• Pascal, COBOL, FORTRAN • Basic, Ada
Nível Médio	• C, C++ • FORTH
Nível Baixo	• Assembly

- Permite acesso de baixo-nível, através de inclusões de código Assembly no meio do programa C.
- Permite manipulação de bits, bytes e endereços.

Visão Geral – Classificação das Linguagens

POO	Surge o paradigma do desenvolvimento de programas orientado a objetos, e a forma de programar busca diminuir a distância das máquinas aos seres humanos
Programação Modular	Conforme o aumento da complexidade os programas passam a ficar cada vez maiores e necessitam ser agrupados em módulos de funcionalidade
Programação Estruturada	Uso de algoritmos que utilizam estruturas de controle que permitem o desenvolvimento de uma lógica mais elaborada
Programação Linear	O programador tem uma certa independência do hardware, mas ainda é muito limitado.
Programação Baixo Nível	A forma de programar está muito associada as características do hardware

Visão Geral - Características

- Características da Linguagem C
  - Estruturas em blocos - Funções - Alta Portabilidade
  - Poucas restrições - Reaproveitamento de código.

- C é uma Linguagem Compilada. Ou seja, um compilador lê o programa inteiro e converte-o em um *código-objeto* (Código binário ou Código de Máquina).

- C é *"Case Sensitive"* - Distinção na utilização de maiúsculas e minúsculas.
  - As variáveis **SOMA**, **Soma** e **soma** são diferentes.

Visão Geral - Características

- Todo programa em C é formado por uma ou mais funções. A função **main()** é a única que precisa necessariamente estar presente no código.

```
#include <stdio.h>

int a, b, c;

main()
{
    scanf("%d",&a);
    scanf("%d",&b);
    c = a + b;
    printf(" \n a + b = %d", c);
}
```

Visão Geral - Características

- Todo programa em C é formado por uma ou mais funções. A função **main()** é a única que precisa necessariamente estar presente no código.

* Os programas em C são formados basicamente por funções.
  - Facilitam a modularização e passagem de parâmetros entre os módulos.

```
#include <stdio.h>

int a, b, c;

main()
{
    scanf("%d",&a);
    scanf("%d",&b);
    c = a + b;
    printf(" \n a + b = %d", c);
}
```

Visão Geral - Características

- Todo programa em C é formado por uma ou mais funções. A função **main()** é a única que precisa necessariamente estar presente no código.

* As **chaves** (**{}**) são utilizadas para agrupar os blocos de comando.

```
#include <stdio.h>

int a, b, c;

main()
{
    scanf("%d",&a);
    scanf("%d",&b);
    c = a + b;
    printf(" \n a + b = %d", c);
}
```

Visão Geral - Características

- Todo programa em C é formado por uma ou mais funções. A função **main()** é a única que precisa necessariamente estar presente no código.

* O **Ponto e Vírgula** (**;**) utilizado para finalizar um comando.

```
#include <stdio.h>

int a, b, c;

main()
{
    scanf("%d",&a);
    scanf("%d",&b);
    c = a + b;
    printf(" \n a + b = %d", c);
}
```

Visão Geral - Características

- Todo programa em C é formado por uma ou mais funções. A função **main()** é a única que precisa necessariamente estar presente no código.

* Biblioteca Básica (E/S)

#include <stdio.h>

(std = Standard; io = input/output)

```
#include <stdio.h>

int a, b, c;

main()
{
    scanf("%d",&a);
    scanf("%d",&b);
    c = a + b;
    printf(" \n a + b = %d", c);
}
```

Visão Geral – Linguagem C

- Comparando diferentes Níveis de Linguagens

```
Program Sequencial;

Uses crt;

Var a, b, c : integer;

begin
   readln(a);
   readln(b);
   c := a + b;
   writeln(' a + b = ', c);
end.
```

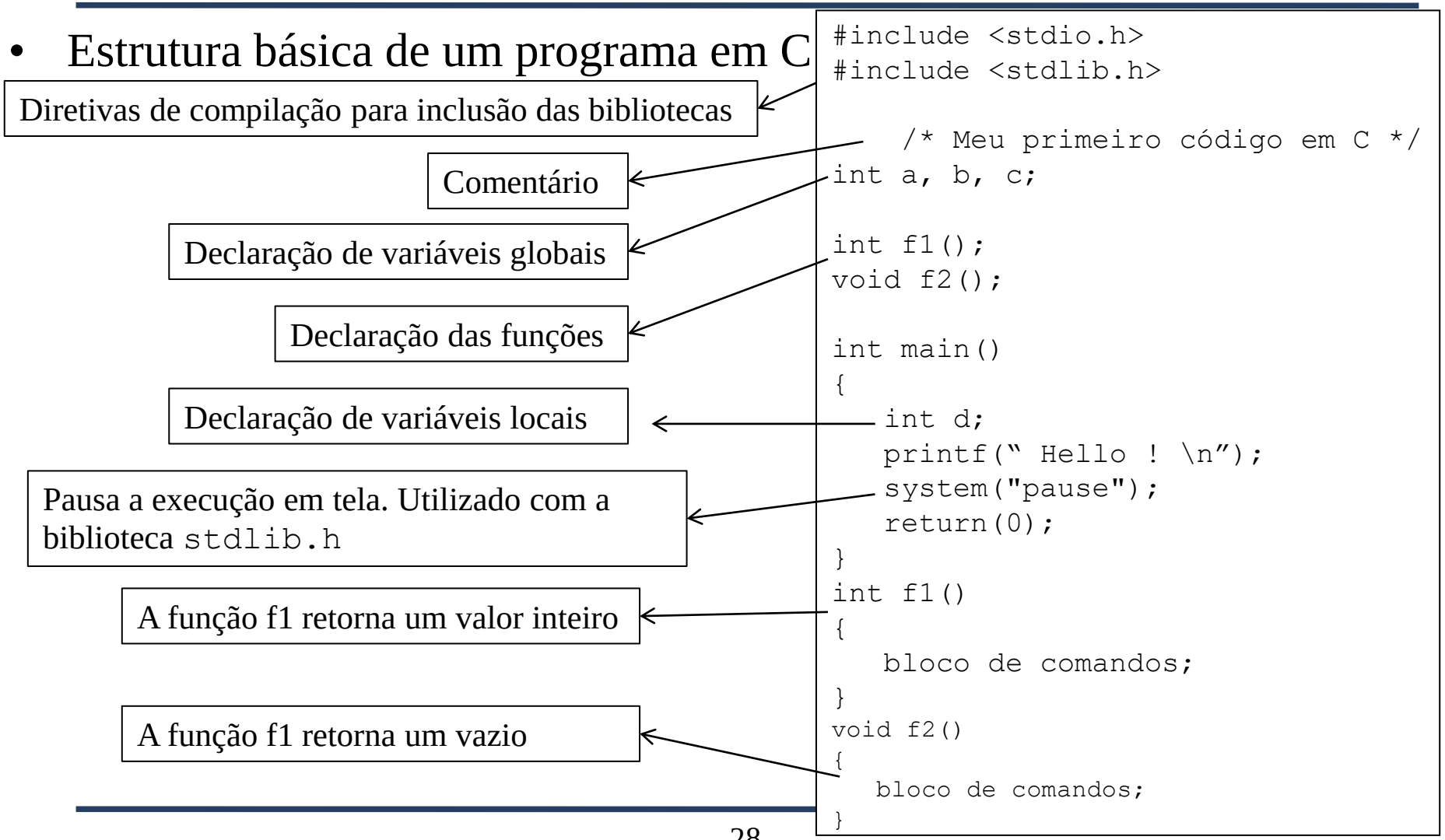
```
#include <stdio.h>

int a, b, c;

main()
{
   scanf("%d",&a);
   scanf("%d",&b);
   c = a + b;
   printf(" \n a + b = %d", c);
}
```

Visão Geral – Linguagem C

- Estrutura básica de um programa em C



Estrutura de Dados – Tipo de Dados

- Podemos dizer que um **tipo de dado** refere-se a um **conjunto de valores** e a um determinado **conjunto de operações** sobre estes valores.
- **Tipos de Dados Básicos**

C
char
int
float / double
int
void

Em C, "Verdadeiro" é qualquer valor diferente de zero.
1 = Verdadeiro ; 0 = Falso

O tipo **void** declara explicitamente uma função que não retorna valor algum.

- **Modificadores de Tipos Básicos**
  - Long (long int ; long double) - Short (short int)
  - Signed - Permite o uso de sinal (signed int) - Unsigned – Não permite o uso de sinal (unsigned int)

Declarações -Variáveis e Constantes

- Declaração de Variáveis - *<tipo> <nome>;*

```
int i, j, k;
unsigned int positivos;
double valor;
char letra;
```

```
int i = 100;
char letra = '2';
char nome[80] = "jose";
float total = 102.30;
```

- Declaração de Constantes – *const <tipo> <nome>;*

```
const int i = 100;
const char letra = '2';
```

- Comando de atribuição – *variável = expressão;*

```
i = 100;
letra = '2';
```

Atribuição Múltipla

```
i = k = 100;
```

- Comandos de Entrada e Saída

```
scanf("%d",&a);
scanf("%d",&b);
c = a + b;
printf(" \n a + b = %d", c);
```

O operador "&" significa "endereço de"
"%d" – formato para leitura e escrita de um inteiro
\n – quebra de linha

Comandos de Entrada e Saída

- Alguns especificadores de formato da função **“printf()”**

Especificador	Tipo do Argumento	Descrição
%d	int	Valor inteiro decimal
%i	int	Valor inteiro decimal
%o	int	Valor inteiro octal
%x	int	Valor inteiro hexadecimal
%X	int	Valor inteiro hexadecimal
%u	int	Valor inteiro decimal sem sinal (*unsigned int*)
%c	int	Um caracter em formato ASCII (código binário correspondente)
%s	char	Uma cadeia de caracteres (*string*) terminada em “**\0**”
%f	float	Valor em ponto flutuante no formato [-]m.dddddd, onde o Nº de d’s é dado pela precisão (padrão é 6)
%e	float ou double	Valor em ponto flutuante em notação exponencial no formato [-]m.ddddd e±xx, onde o Nº de d’s é dado pela precisão
%E	float ou double	Valor em ponto flutuante em notação exponencial no formato [-]m.ddddd E±xx, onde o Nº de d’s é dado pela precisão
%g	float ou double	Valor em ponto flutuante no formato %e (quando o expoente for menor que -4 ou igual a precisão) ou %f (nos demais casos). Zeros adicionais e um ponto decimal final não são impressos
%G	float ou double	Valor em ponto flutuante no formato %E (quando o expoente for menor que -4 ou igual a precisão) ou %f (nos demais casos). Zeros adicionais e um ponto decimal final não são impressos
%%	-	Exibe o caracter ‘%’

Comandos de Entrada e Saída

- Alguns especificadores de formato da função **“scanf()”**

Especificador	Descrição
%d	Indica um valor inteiro decimal
%i	Indica um valor inteiro (podendo estar na base decimal, octal com inicial **0** (zero) ou hexadecimal com inicial **0X**
%u	Indica um valor inteiro decimal sem sinal
%c	Indica um caracter (*char*)
%s	Indica uma *string*
%f, %e, %g	Indica um valor em ponto flutuante de precisão simples (*float*)
%lf, %le, %lg	Indica um valor em ponto flutuante de precisão dupla (*double*)
%o	Indica um valor inteiro octal (com ou sem zero inicial)
%x ou %X	Indica um valor inteiro hexadecimal (com ou sem “0x” inicial)

Operadores

- Comparação entre os Operadores Básicos

C	Operação	
+	Adição	Operadores Aritméticos
-	Subtração	
*	Multiplicação	
/	Divisão	
%	Resto	
<	Menor que	Operadores Relacionais
<=	Menor que ou Igual	
>	Maior que	
>=	Maior que ou Igual	
= =	Igual	
!=	Diferente	
!	Negação	Operadores Lógicos
&&	Conjunção	
||	Disjunção	

Estruturas de Controle

- Sequencial

```
instrução1;
instrução2;
instrução3;
...;
...;
```

```
#include <stdio.h>

int a, b, c;

main()
{
   scanf("%d",&a);
   scanf("%d",&b);
   c = a + b;
   printf(" \n a + b = %d", c);
}
```

Estruturas de Controle

- Condicional

```
if (condição)
   instrução1;
```

```
if (condição)
   instrução1;
else
   instrução2;
```

if - else

```
#include <stdio.h>
int a, b, c;
main()
{
   scanf("%d",&a);
   scanf("%d",&b);
   if (a < b)
   {
     printf(" \n a é menor ”);
     printf(" \n b é maior ”);
   }
   else
   {
     printf(" \n b é menor ”);
     printf(" \n a é maior ”);
   }
}
```

```
if (condição)
{
   instrução1;
   instrução2;
}
```

```
if (condição)
{
   instrução1;
   instrução2;
}
else
{
   instrução3;
   instrução4;
}
```

Estruturas de Controle

- Repetição

```
while (condição)
{
   instrução1;
   instrução2;
}
```

while

```
#include <stdio.h>
int k;
main()
{
    k = 0;
    while (k <= 100)
    {
      printf(" \n %d ", k);
      k = k + 2;
    }
}
```

```
for (inicialização; condição; incremento)
{
   instrução1;
   instrução2;
}
```

for

```
#include <stdio.h>
main()
{
  for (int k = 0; k <= 100; k=k+2)
  {
      printf(" \n k = %d", k);
   }
}
```

Exercício

Problemas

02 - Elabore um algoritmo que verifique se um dado número inteiro positivo é par ou impar.

03 – Faça um algoritmo que leia 2 valores numéricos e um símbolo. Caso o símbolo seja um dos relacionados abaixo efetue a operação correspondente com os valores.

“+”“-” “*” “/”

Problemas – Estruturas Condicionais

04 – Faça um algoritmo que calcule a média ponderada de um aluno, a partir de suas 3 notas obtidas no curso. Sabendo-se que a primeira avaliação tem peso 2, a segunda tem peso 4, a terceira tem peso 4.
Mostre ao final a mensagem:
"A MEDIA FINAL DE .....FOI .....". Informar também se o aluno foi aprovado, Mostrando a mensagem "APROVADO", caso a nota final seja maior ou igual a 7,0.

05 – Modifique a questão anterior para informar :

APROVADO	Caso a nota final seja entre 7 e 10
RECUPERAÇÃO	Caso a nota final seja entre 5 e 7
REPROVADO	Caso a nota final seja entre 0 e 5.

06 – Fazer um algoritmo que leia 3 valores inteiros e escreva o menor deles.

Problemas – Estruturas de Repetição

07 – Faça um programa para mostrar os números de 1 a 100

08 – Faça um programa para ler 10 números e apresentar o resultado da soma destes números

09 – Faça um programa que imprima os números inteiros entre um dado intervalo A e B.

Problemas – Estruturas de Repetição

10 – A partir da leitura de um número indeterminado de valores não nulos, determinar e exibir quantos desses valores são positivos e quantos são negativos. Determinar também a soma dos positivos e a soma dos negativos. O último valor a ser lido é zero (0 é o flag).

11 – A partir da leitura de um numero indeterminado de notas, calcular e exibir a média aritmética dessas notas. O flag para indicar o fim das notas é –1.

12 – O mesmo exercício anterior fornecendo adicionalmente a maior e a menor nota lida.