# Linguagem C

Prof. Sidney Rodrigues Cunha

## Fundamentos e Princípios

Revisão 2011.1

# Linguagem C

## Algorítimos

**Exemplo: trecho de um algoritmo escrito em Pseudo-linguagem que recebe um número num e escreve a tabuada de 1 a 10 para este valor:**

leia num

para n de 1 até 10 passo 1 faça

tab ← num * n

imprime tab

fim faça

# Linguagem C

Algorítimos

**Exemplo: trecho do mesmo programa escrito em linguagem C:**

```
scanf(&num);

for(n = 1; n <= 10; n++){

tab = num * n;

printf("\n %d", tab);

};
```

# Linguagem C

## Algorítimos

**Exemplo: trecho do mesmo programa escrito em linguagem Basic:**

```
10 input num

20 for n = 1 to 10 step 1

30 let tab = num * n

40 print chr$ (tab)

50 next n
```

# Linguagem C

Algorítimos

**Exemplo: trecho do mesmo programa escrito em linguagem Fortran:**

```
read (num);

do 1 n = 1:10

tab = num * n

write(tab)

10 continue
```

# Linguagem C

Algorítimos

**Exemplo: trecho do mesmo programa escrito em linguagem Assembly para INTEL 8088:**

```
MOV CX,0
IN AX,PORTA
MOV DX,AX
LABEL:
INC CX
MOV AX,DX
MUL CX
OUT AX, PORTA
CMP CX,10
JNE LABEL
```

# Linguagem C

**Linguagens de baixo nível...**

São linguagens voltadas para a máquina, isto é, são escritas usando as instruções do microprocessador do computador. São genericamente chamadas de linguagens *Assembly*.

*Vantagens: Programas são executados com maior velocidade de processamento. Os programas* ocupam menos *espaço na memória.*

*Desvantagens: Em geral, programas em Assembly tem pouca portabilidade, isto é, um código* gerado para um tipo de processador não serve para outro.

Códigos Assembly não são estruturados, tornando a *programação mais difícil.*

# Linguagem C

**Linguagens de alto nível:**

São linguagens voltadas para o ser humano. Em geral utilizam sintaxe estruturada tornando seu código mais legível. Necessitam de *compiladores ou interpretadores para gerar* instruções do microprocessador.

Interpretadores fazem a interpretação de *cada instrução do programa* fonte executando-a dentro de um ambiente de programação, **Basic e AutoLISP por exemplo.**

Compiladores fazem a tradução de *todas as instruções do programa fonte gerando um programa* executável. Estes programas executáveis (*.exe) podem ser executados fora dos ambientes de programação, **C e Pascal** por exemplo. As linguagens de alto nível podem se distinguir quanto a sua aplicação em termos *como* ***C, Pascal e Basic ou específicas como Fortran (cálculo matemático),*** **GPSS (simulação), LISP (inteligência artificial) ou CLIPPER (banco de dados).**

# Linguagem C

***Linguagens de Alto Nível***

***Vantagens:*** *Por serem compiladas ou interpretadas, tem maior portabilidade podendo serem* executados em varias plataformas com pouquíssimas modificações. Em geral, a programação torna-se mais fácil por causa do maior ou menor grau de estruturação de suas linguagens.

***Desvantagens:*** *Em geral, as rotinas geradas (em linguagem de maquina) são mais genéricas* e portanto mais complexas e por isso são mais lentas e ocupam mais memória.

# Linguagem C

Conceito ...

A **linguagem C** é uma linguagem de *alto nível, genérica. Foi desenvolvida por programadores para programadores tendo como meta características de flexibilidade e portabilidade. O C é uma* linguagem que nasceu juntamente com o advento da teoria de *linguagem estruturada e do computador pessoal. Assim tornou-se rapidamente uma linguagem "popular" entre os programadores. O C foi usado* para desenvolver o sistema operacional UNIX, e hoje esta sendo usada para desenvolver novas linguagens, entre elas a linguagem C++ e Java.

# Linguagem C

O **C** é uma linguagem de alto nível com uma sintaxe bastante estruturada e flexível tornando sua programação bastante simplificada.

• Programas em C são compilados, gerando programas executáveis.

• O C compartilha recursos tanto de alto quanto de baixo nível, pois permite acesso e programação direta do microprocessador. Com isto, rotinas cuja dependência do tempo é crítica, podem ser facilmente implementadas usando instruções em Assembly. Por esta razão o C é a linguagem preferida dos programadores de aplicativos.

• O **C** é uma linguagem estruturalmente simples e de grande portabilidade. O compilador C gera códigos mais enxutos e velozes do que muitas outras linguagens.

• Embora estruturalmente simples (poucas funções intrínsecas) o **C** não perde funcionalidade pois permite a inclusão de uma farta quantidade de rotinas do usuário. Os fabricantes de compiladores fornecem uma ampla variedade de rotinas pré-compiladas em bibliotecas.

# Linguagem C

Fases da compilação

1. Análise sintática (e léxica)
2. Análise contextual
3. Geração de código

Análise léxica – lê a seqüencia de caracteres e a organiza como *tokens* – seqüencias de caracteres com algum significado.

Análise sintática – agrupa caracteres ou *tokens* em uma estrutura hierárquica com algum significado.

Geração do Código – Um tradutor traduz os grupos de caracteres em código de máquina.

# Linguagem C

Estrutura de um programa em C Um programa em C é constituído de:

Um cabeçalho contendo as diretivas de compilador onde se definem o valor de constantes simbólicas, declaração de variáveis, inclusão de bibliotecas, declaração de rotinas, etc.

Um bloco de instruções principal e outros blocos de rotinas.

Documentação do programa: comentários.

# Linguagem C

**Conjunto de caracteres**

Um programa fonte em C é um texto **não** formatado escrito em um editor de textos usando um o conjunto padrão de caracteres ASCII.

**Caracteres válidos:**

a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1 2 3 4 5 6 7 8 9 0
+ - * / \ = | & ! ? # % ( ) { } [ ] _ ‘ “ . , : < >

**Caracteres não válidos:**

@ $ ¨ á é õ ç

**Os caracteres acima são válidos apenas em strings.**

# Linguagem C

Comentários

Em C, comentários podem ser escritos em qualquer lugar do texto para facilitar a interpretação do algoritmo. Para que o comentário seja identificado como tal, ele deve ter um /* antes e um */ depois.

**Exemplo:**
/* esta e´ uma linha de comentário em C */

**Observação: O** C++ permite que comentários sejam escritos de outra forma: colocando um // em uma linha, o compilador entenderá que tudo que estiver a direita do símbolo é um comentário.

**Exemplo:**
// este e´ um comentário valido apenas em C++

# Linguagem C

**Diretivas de Compilação**

Em C, existem comandos que são processados durante a compilação do programa. Estes comandos são chamados de ***diretivas de compilação***. *Estes comandos informam ao* compilador do C basicamente quais são as constantes simbólicas usadas no programa e quais bibliotecas devem ser anexadas ao programa executável.

A diretiva **#include** diz ao compilador para incluir na compilação do programa outros arquivos. Geralmente estes arquivos contem bibliotecas de funções ou rotinas do usuário.

A diretiva **#define** diz ao compilador quais são as constantes simbólicas usadas no programa.

# Linguagem C

**Declaração de variáveis**

Em C, como na maioria das linguagens, as variáveis devem ser declaradas no inicio do programa.Estas variáveis podem ser de vários tipos:

int (inteiro), float (real de simples precisão) No exemplo anterior num, raiz, inf e sup são declaradas como variáveis reais, enquanto i é declarada como uma variável inteira.

# Linguagem C

**Entrada e saída de dados**

Em C existem varias maneiras de fazer a leitura e escrita de informações. Estas operações são chamadas de operações de entrada e sadia. Outras funções de leitura e escrita em arquivos, saída gráfica, funções de manipulação de *mouse, entrada e saída de informações via portas serial e paralela e etc.*

**Exemplo:**

O comando **printf** é uma função de escrita na tela, **scanf** é uma função de leitura de teclado.

# Linguagem C

**Estruturas de controle**

A linguagem C permite uma ampla variedade de estruturas de controle de fluxo de processamento.

**Estrutura de Decisão:**

Permite direcionar o fluxo lógico para dois blocos distintos de instruções conforme uma condição de controle.

# Linguagem C

Estruturas básicas (decisão e repetição) são muito semelhantes as estruturas usadas nas Pseudo-linguagem algorítmicas:

**Pseudo-linguagem**

se *condição*
então *bloco 1*
senão *bloco 2*
fim se

**Linguagem C**

```
if(condição) {
 bloco 1;
}else {
 bloco 2;
} ;
```

# Linguagem C

**Estrutura de Repetição:**

Permite executar repetidamente um bloco de instruções ate que uma condição de controle seja satisfeita.

| Pseudo-linguagem | Linguagem C |
| --- | --- |
| **faça**<br>*bloco*<br>**até *condição*** | **do{**<br>*bloco;*<br>***}while (condição);*** |

# Linguagem C

**Constantes**

1. inteiras
2. de ponto flutuante
3. caracteres
4. *strings*.

Dica do dia...

Constantes inteiras e de ponto flutuante representam números de um modo geral. Caracteres e strings representam letras e agrupamentos de letras (palavras).

# Linguagem C

**Constantes inteiras**

Uma constante inteira é um número de valor inteiro. De uma forma geral, constantes inteiras são seqüências de dígitos que representam números inteiros. Números inteiros podem ser escritos no formato decimal (base 10), hexadecimal (base 16) ou octal (base 8).

Uma constante inteira decimal é formada por uma seqüência de dígitos decimais: 0, 1, 2,3, 4, 5, 6, 7, 8, 9. Se a constante tiver dois ou mais dígitos, o primeiro não pode ser 0. Sendo 0 o compilador considerará esta constante como octal e não decimal.

**Exemplo:**

Constantes inteiras decimais válidas.

0 3 -45 26338 -7575 1010

# Linguagem C

**Exemplo:**

Constantes inteiras decimais inválidas.

1. (ponto)
1,2 (vírgula)
045 (primeiro dígito é 0: não é constante decimal)
212-22-33 ( caracter ilegal: - )

Uma constante inteira hexadecimal é formada por uma seqüência de dígitos decimais: 0, 1,2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F (ou a, b, c, d, e). Uma constante hexadecimal deve começar por 0x. Neste caso, os dígitos hexadecimais podem ser minúsculos ou maiúsculos.

**Exemplo:**

Constantes inteiras hexadecimais válidas.

0x0 0x3 0x4f5a 0x2FFE OxABCD 0xAaFf

# Linguagem C

**Exemplo:**

Algumas constantes inteiras hexadecimais inválidas.

0x3. (ponto)
0x1,e (vírgula)
0x ff (espaço)
FFEE (não começa com 0x: não é constante hexadecimal)
0Xfg34 (caracter ilegal: g)

Constante Octal

Uma constante inteira octal é formada por uma seqüência de dígitos octais: 0, 1, 2, 3, 4, 5, 6, 7. A constante octal deve ter o primeiro digito 0 para que o compilador a identifique como tal.

# Linguagem C

**Exemplo:**

Constantes octais válidas.

00 -03 045 02633 07575 -0101

**Exemplo:**

Constantes inteiras octais inválidas.

010. (ponto)
01,2 (vírgula)
0 4 (espaço)
45 (primeiro digito não é 0: não é constante octal)
01784 (caracter ilegal: 8)

# Linguagem C

**Constantes de ponto flutuante**

Números reais (não inteiros) são representados em base 10, por números com um ponto decimal e (opcionalmente) um expoente.

Um número ponto flutuante deve ter um ponto decimal que não pode ser substituído por uma vírgula.

Um número de ponto flutuante pode ser escrito em notação científica. Neste caso o x10 é substituído por e ou E. O número 1.23e4 representa $1.23 \times 10^4$ ou 12300.

**Exemplo:**

Números de ponto flutuante válidos.

0.234 125.65 .93 1.23e-9 -1.e2 10.6e18 -.853E+67

# Linguagem C

Números Reais, relembrando...

Os números reais são números usados para representar uma quantidade contínua (incluindo o zero e os negativos).

Literalmente...

O conjunto dos números reais  é uma expansão do conjunto dos números racionais que engloba não só os inteiros e os fracionários, positivos e negativos, mas também todos os números irracionais.

# Linguagem C

**Exemplo:**

O número 314 pode ser representado por qualquer uma das seguintes formas:

314. 3.14e2 +3.14e+2 31.4e1 .314E+3 314e0

# Linguagem C

**Constantes caracteres**

Uma constante caracter é uma letra ou símbolo colocado entre aspas simples.

**Exemplo:**

Algumas constantes caracteres.

'a' 'b' 'X' '&' '{' ' '

Embora sejam visualizados como letras e símbolos as constantes caracteres são armazenadas internamente pelo computador como um número inteiro entre 0 e 255. O caracter 'A' por exemplo, tem valor 65. Os valores numéricos dos caracteres estão padronizados em uma tabela chamada de *American Standard Code for Information Interchange Table ou simplesmente tabela ASCII*.

# Linguagem C

**Exercícios**