


UNIVERSIDADE
FUMEC

■ Algoritmos e Estrutura ■ de Dados

Videoaula 01 (Parte 1/3)

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG




UNIVERSIDADE
FUMEC

Uma visão geral da disciplina

Algoritmos e Estrutura de Dados

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

2



UNIVERSIDADE
FUMEC

Objetivo

O conhecimento de linguagens de programação, por si só, **não capacita programadores** ...

“é necessário **saber usá-las** de **maneira eficiente**.”

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

3



Objetivo

Ao final desta disciplina o aluno será capaz de desenvolver programas ...

... com uma **representação adequada dos dados** em vista das **funcionalidades** que devem ser atendidas e ...

Terá uma visão de como se construir programas de nível médio-avançado relativamente **eficientes** e **confiáveis**.



O que veremos?



Cronograma

1ª Aula

- Conceitos Fundamentais + Expressões.

2ª Aula

- Controle de Fluxo + Funções.

3ª Aula

- Vetores e Alocação Dinâmica + Matrizes.

4ª Aula

- Cadeias de Caracteres + Tipos Estruturados.

Cronograma

5ª Aula

- Tipos Abstratos de Dados + Listas.

6ª Aula

- Pilhas + Filas.

7ª Aula

- Árvores.

8ª Aula

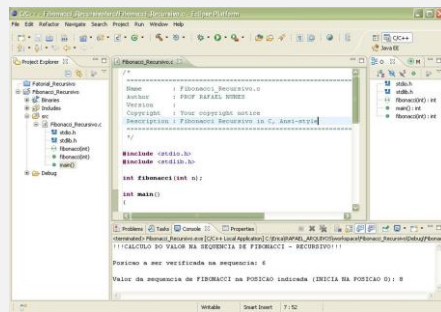
- Revisão do Conteúdo.

A ferramenta

Eclipse CDT

<https://eclipse.org/cdt/>

Eclipse CDT



Você deve **preparar-se** para dedicar um esforço extra com esta disciplina, uma vez que não se trata de uma matéria trivial.

Objetivos de Aprendizagem

- ☐ Familiaridade com os **conceitos básicos** da Linguagem de Programação C.
- ☐ Linguagem C – **Expressões:**
 - Variáveis.
 - Operadores.

Uma breve Introdução!

Quando e como surgiu a **Linguagem C**?

Histórico



- A linguagem C surgiu na década de 70.
- Seu inventor, **Dennis Ritchie**, implementou-a pela primeira vez usando um computador **DEC PDP-11** rodando o sistema operacional **UNIX**.

Histórico

- O C é derivado de uma outra linguagem:
A **linguagem B**, criada por Ken Thompson.
- O B, por sua vez, veio da **linguagem BCPL**, inventada por Martin Richards.



Onde podemos utilizar a Linguagem C?

Aplicação da Linguagem C

O C é uma linguagem de programação genérica que é utilizada para a **criação de programas** diversos. Então podemos encontrá-la em:

- ☐ processadores de texto (ex.: Vi, Vim);
- ☐ sistemas operacionais (ex.: UNIX, Linux);
- ☐ e até mesmo na programação de GAMES!

http://www.nvidia.com/object/cuda_home_new.html

A Linguagem C

Conceitos Básicos - Parte 1

A linguagem C é:

Case Sensitive

**Letras Maiúsculas e Minúsculas
fazem diferença!**

Case Sensitive

- ☐ **Maiúsculas** e **Minúsculas** fazem diferença!
- ☐ Se declararmos uma variável com o nome de resultado ela será diferente das declarações abaixo:

- Resultado.
- RESULTADO.
- ReSuLtAdO, e;
- rEsUiTaDo.



resultado

Case Sensitive

- ☐ Da mesma maneira, os comandos da linguagem C, como **if** e **for**, devem ser escritos sempre em letra minúscula.

**E o que acontece se os comandos da
linguagem forem escritos de maneira
diferente?**

- ☐ O compilador não irá interpretá-los como sendo comandos, mas sim como variáveis.

Entendendo *a estrutura* de um programa em C

Estrutura de um Programa em C

O programa deve estar organizado nas seguintes partes:

1. **Cabeçalho:**
 - Funções Externas ao seu programa.
 - Comandos para o Pré-processador.
 - Variáveis Globais.
2. **Funções** ou apenas os **protótipos das funções**.
3. **Função Principal:**
 - `main ()`
4. **Implementação das funções** do seu programa.

Analizando meu *Primeiro Programa* em C



Programa 01 – alomundo.c

Abram a ferramenta e após criar um novo projeto digitem o seguinte programa abaixo:

```
01  /* Meu Primeiro Programa */
02  #include <stdio.h>
03  int main (void)
04  {
05      //Imprime a seguinte mensagem na tela
06      printf ("Ola! Eu nasci!\n");
07
08      return 0;
09
10  } //Fim
```



Os comentários na Linguagem C

*Linhas 01, 05 e 10
do programa “alomundo.c”*



Comentários na Linguagem C

- ☐ Comentário indica um texto que **não** será verificado pelo tradutor (Compilador), ou seja, você poderá escrever qualquer coisa.
- ☐ Geralmente deve ser utilizado para descrever **o que faz o seu código** ao longo do programa.

Existem **dois tipos** de comentários na linguagem:

- Comentário **de Linha**.
- Comentário **de Bloco**.

Comentário de Linha

*Linhas 05 e 10
do programa "alomundo.c"*

Comentário de Linha

Abrange apenas **uma linha**. Para escrever um comentário de linha faça:

- Digite duas vezes a barra p/ direita do teclado ("//") no lugar onde você deseja inserir um comentário. Veja abaixo:

```
01 //Este é um comentário de linha
02 //Este também é um comentário
03 int main () {
04     ...
05 } //Opa! Eu também sou um comentário
```

Comentário de Bloco

*Linha 01
do programa "alomundo.c"*



Comentário de Bloco

Abrange todo **um bloco** e não apenas uma linha.

Para escrever um bloco de comentário:

1. Inicie o bloco com uma barra p/ direita ("**/**") seguida por um asterisco ("*****")
2. Termine o bloco com o inverso, um asterisco ("*****") seguido por uma barra para a direita ("**/**")

```
01 /* Este é um comentário de bloco
02  é utilizado quando precisamos comentar
03  mais de uma linha em sequência */
04 int main () {
05     ...
06 }
```



Diretiva de Compilação

Linha 02
do programa "alomundo.c"



Diretiva de Compilação

- ☐ Uma **diretiva de compilação** é uma instrução para o compilador.
- ☐ O primeiro caractere da linha deve ser um **"#"**
- ☐ Algumas diretivas de compilação:
 - **include**: inclui o arquivo indicado.
 - **define**: define uma macro.
 - **if, elif, else, endif**: compilação condicional.



Diretiva de Compilação

- ❑ A linha `#include <stdio.h>` diz ao compilador que ele deve incluir o arquivo-cabeçalho `stdio.h`.
- ❑ Neste arquivo existem declarações de *funções* úteis para *entrada e saída* de dados:
 - `std` = standard, padrão em inglês.
 - `io` = Input/Output, entrada e saída.
 - `stdio` = Entrada e saída padronizadas.
- ❑ O C possui *diversos* arquivos-cabeçalhos (*Bibliotecas*).



Função Principal

*Linha 03
do programa "alomundo.c"*



Função Principal

- ❑ A linha 3 do "programa1" contém o início da *função main()*.
- ❑ Esta função marca o *início da execução do programa* e deve existir em algum lugar do código para este funcionar corretamente.
- ❑ Se o seu programa tiver somente uma função, ela deverá ser *main()*.

O que significa aquele *int* imediatamente antes da função principal *main()*?

int *main()*

- ☐ Este *int* está dizendo que *main()* deve retornar um valor para o sistema, indicando *sucesso ou falha*.
- ☐ Este valor é um número *inteiro*, por isso *int*.
- ☐ Veremos mais tarde sobre tipos de dados, e também sobre funções.

E as chaves “{}” após a função principal?

*Linhas 04 e 10
do programa “alomundo.c”*



Delimitadores de bloco de Código

- ❑ Na linha 4 vemos o caractere *abre-chaves*, um delimitador de bloco “{”
 - Início do bloco.
- ❑ Na linha 10 vemos o caractere *fecha-chaves*, outro delimitador de bloco “}”
 - Fim do bloco.
- ❑ Neste caso, o que estiver dentro do *par de chaves* pertence a função main().



printf()

*Linha 06
do programa “alomundo.c”*



A função printf()

- ❑ Na *linha 6* que o nosso programa cumpre a sua crucial missão:
 - Escrever a mensagem *“Ola! Eu nasci!”* na tela.
- ❑ A função *printf()* é a principal função para escrita no console (saída padrão).

E o “\n” dentro da função printf() ?

*Linha 06
do programa “alomundo.c”*

Caractere de nova linha – “\n”

- ☐ O “\n” é um caractere especial, e serve para *pular para a próxima linha*, assim que acabar de escrever a mensagem.
- ☐ Iremos estudar esses caracteres especiais posteriormente.

Para reflexão!

Você conseguiria *explicar* todas as linhas do programa apresentado novamente?



No próximo bloco...

Veremos sobre:

- A linguagem C.
- Os tipos da Linguagem C.
- Os *Modificadores* de Tipo.
- Declaração de Variáveis, etc.

Bons estudos!



Algoritmos e Estrutura de Dados

Videoaula 01 (Parte 2/3)

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG



A Linguagem C

Variáveis



Nomes de Variáveis

As variáveis no C podem ter qualquer nome se **duas condições forem satisfeitas**:

- ☐ o nome deve **começar** com **uma letra** ou **sublinhado** (`_`).
- ☐ os caracteres **subseqüentes** devem ser **letras**, números ou **sublinhado** (`_`).



Nomes de Variáveis

Há apenas mais duas restrições:

- ☐ o nome de uma variável **não** pode ser igual a uma **palavra reservada**.
- ☐ **_nem** igual ao **nome de uma função** declarada pelo programador, ou pelas bibliotecas do C.



Nomes de Variáveis

- ☐ Variáveis de **até 32 caracteres** são aceitas.
- ☐ É sempre bom lembrar que o C é "**case sensitive**" e portanto deve-se prestar atenção às letras **maiúsculas** e **minúsculas**.

Os *Tipos* da Linguagem C

Tipos da Linguagem C

O C tem 5 tipos básicos:

- ☐ char;
- ☐ int;
- ☐ float;
- ☐ void;
- ☐ double.

Os *Modificadores* de Tipo

Modificadores de Tipo

Para cada um dos tipos de variáveis existem os **modificadores de tipo**.

Os modificadores de tipo do C **são quatro**:

- ☐ signed;
- ☐ unsigned;
- ☐ long;
- ☐ short.

Modificadores de Tipo

- ☐ Ao **float** **não** se pode aplicar nenhum modificador.
- ☐ Ao **double** aplica-se **apenas o long**.
- ☐ Os **quatro** podem ser aplicados a **inteiros**.
- ☐ A intenção é que **short** e **long** devam prover **tamanhos diferentes de inteiros** onde isto for prático.

Tipos + Modificadores de Tipo

Tipo	Num de bits	Intervalo	
		Início	Fim
char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127
int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
float	32	3,4E-38	3,4E+38
double	64	1,7E-308	1,7E+308
long double	80	3,4E-4932	3,4E+4932

Observações



- ❑ O tipo **long double** é o tipo de ponto flutuante com maior precisão.
 - ❑ É importante observar que os intervalos de ponto flutuante, na tabela anterior, estão indicados em **faixa de expoente...**
- ... mas os números podem assumir valores tanto **positivos** quanto **negativos**.

Declaração de Variáveis



Declaração de Variáveis



- ❑ As variáveis no C devem ser declaradas antes de serem usadas.
- ❑ A forma geral da declaração de variáveis é:

<tipo_da_variável> <nome_da_variável>

Exemplo: **int x;**



Declaração de Variáveis

- ❑ O **tipo default do C** é o **int**, se colocarmos um modificador sem o tipo, ele sempre será int.
- ❑ Assim um **long** sozinho será equivalente a um **long int**.

Veja exemplo:

1. **char** ch, letra;
2. **long** cont; **//long int** cont
3. **float** pi;



Declaração de Variáveis

Há três lugares nos quais podemos declarar variáveis:

O primeiro é **fora de todas as funções** do programa.

- Chamadas de **variáveis globais**.
- Podem ser utilizadas a partir de **qualquer lugar no programa**.
- Devem ser utilizadas com cuidado! (Inconsistência).



Declaração de Variáveis

O segundo é no início de um **bloco de código**.

- ❑ Chamadas de **variáveis locais**.
- ❑ Elas **só têm validade dentro do bloco** no qual são declaradas.

... isto é, **somente a função à qual ela pertence** reconhece a existência desta variável.

Declaração de Variáveis



- ❑ O terceiro lugar onde se pode declarar variáveis é na **lista de parâmetros de uma função**.
- ❑ Apesar destas variáveis receberem **valores externos**, elas são conhecidas **apenas** pela função onde são declaradas.
- ❑ Veremos funções mais adiante nesta disciplina.

Declaração de Variáveis



```

1. #include <stdio.h>
2. int contador;
3. int func1(int j) {
4.     ...
5. }
6. int main()
7. {
8.     char condicao;
9.     int i;
10.    for (i=0; ...)
11.    {
12.        float f2;
13.        ...
14.        func1(i);
15.    }
16.    ...
17.    return(0);
18.}
```

Para reflexão...



Você conseguiria **citar** todos os tipos e modificadores de tipo da linguagem C?



No próximo bloco...

Veremos sobre:

- Operadores Aritméticos e de Atribuição.
- Operadores de *Incremento e Decremento*.
- Operadores *Relacionais e Lógicos*.

Bons estudos!



Algoritmos e Estrutura de Dados


Videoaula 01 (Parte 3/3)

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG



A Linguagem C


Operadores



Operadores Aritméticos e de Atribuição

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

70



Operadores Aritméticos


Os operadores aritméticos são usados para desenvolver operações matemáticas.

A seguir apresentamos a lista dos operadores aritméticos do C:

Operador	Ação
+	Soma (inteira e ponto flutuante)
-	Subtração ou Troca de sinal (inteira e ponto flutuante)
*	Multiplicação (inteira e ponto flutuante)
/	Divisão (inteira e ponto flutuante)
%	Resto de divisão (de inteiros)
++	Incremento (inteiro e ponto flutuante)
--	Decremento (inteiro e ponto flutuante)

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

71



Operadores Aritméticos

Qual seria o resultado **das operações** do trecho de código abaixo:

- int a = 17, b = 3;
- int x, y;
- float z = 17. , z1, z2;
- x = a / b;
- y = a % b;
- z1 = z / b;
- z2 = a/b;

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

72



Operadores Aritméticos

Resultado:

$x = 5$, $y = 2$, $z1 = 5.666666$ e $z2 = 5.0$

Note que na linha correspondente a $z2$, primeiramente é feita uma **divisão inteira** (pois os dois operandos são inteiros).

Somente após efetuada a divisão é que o resultado é atribuído a uma variável float.



Operadores de Incremento e Decremento



Incremento x Decremento

Os operadores de incremento e decremento são **unários** que alteram a variável sobre a qual estão aplicados.

O que eles fazem é **incrementar ou decrementar**, a variável sobre a qual estão aplicados, de **1**.

Então:

$x++$;

$x--$;

... são equivalentes a:

$x = x + 1$;

$x = x - 1$;



Incremento x Decremento

- ❑ Estes operadores podem ser **pré-fixados** ou **pós-fixados**.
- ❑ A diferença é que quando são **pré-fixados** eles incrementam e retornam o valor da variável **já incrementada**.
- ❑ Quando são **pós-fixados** eles retornam o valor da variável **sem o incremento** e depois incrementam a variável.

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

76



Incremento x Decremento

Então, em:

`x=23;`

`y=x++;`

... teremos, no final, **y=23** e **x=24**.

Porém em:

`x=23;`

`y=++x;`

... teremos, no final, **y=24** e **x=24**.

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

77



Operador de Atribuição

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

78



Operador de Atribuição

O operador de atribuição do C é o **=** (igual).

- O que ele faz é pegar o valor à direita e **atribuir à variável da esquerda**.
- Além disto ele **retorna o valor** que ele atribuiu, fazendo com que as seguintes expressões sejam válidas:

1. **x=y=z=1.5;** /* Expressao 1 */

2. **if (k=w) ...** /* Expressao 2 */



Operador de Atribuição

☐ A **expressão 1** é válida, pois quando fazemos $z=1.5$ ela retorna 1.5, que é passado adiante.

☐ A **expressão 2** será verdadeira se w for diferente de zero, pois este será o valor retornado por $k=w$.

Atenção! Pense bem antes de usar a **expressão 2**, pois ela pode gerar erros de interpretação.

- Você não está comparando k e w .
- Você está atribuindo o valor de w a k e usando este valor para tomar a decisão.



Simplificação de Expressões

Podemos **simplificar** algumas **expressões** aplicando o seguinte conceito a todos os operadores aritméticos:

1. $x += 10$ equivale a $x = x + 10$

2. $y -= 20$ equivale a $y = y - 20$

3. $z *= y$ equivale a $z = z * y$

4. $k /= 2$ equivale a $k = k / 2$

Operadores *Relacionais e Lógicos*

Operadores Relacionais

Os operadores relacionais do C *realizam comparações* entre variáveis.

São eles:

`==` (*igual*),
`!=` (*diferente de*),
`>` (*maior que*),
`<` (*menor que*),
`>=` (*maior ou igual*),
`<=` (*menor ou igual*).

Os operadores relacionais retornam verdadeiro (1) ou falso (0).

Operadores Lógicos

Para fazer *operações com valores lógicos* (verdadeiro e falso) temos os operadores lógicos:

`&&` (*and (e)*),
`||` (*or (ou)*),
`!` (*not (não)*).



Operadores Lógicos

Usando os operadores relacionais e lógicos podemos realizar diversos **testes**.

A **tabela-verdade** destes operadores é dada a seguir:

p	q	p AND q	p OR q
falso	falso	falso	falso
falso	verdadeiro	falso	verdadeiro
verdadeiro	falso	falso	verdadeiro
verdadeiro	verdadeiro	verdadeiro	verdadeiro



Operadores Lógicos - Exemplo

No trecho de programa abaixo o comando if será executado, pois o resultado da expressão lógica é verdadeiro:

```
1. int i = 5, j = 7;
2. if ( (i > 3) && (j <= 7) && (i != j) ) j++;
   ( V AND V AND V )
   Resultado: V
```



Para reflexão...

Você conseguiria **explicar**
a diferença entre incremento
pós e pré-fixado?

Finalizando...



Não deixe de interagir no ambiente virtual,
pois ele é nosso meio de comunicação.

Bons estudos e até a próxima videoaula!

Prof. Rafael Nunes – M.Sc. em Engenharia de Computação pela UFMG

88

Referências

Waldemar Celes, Renato Cerqueira, José Lucas
Rangel,. Introdução a Estruturas de Dados, Editora
Campus. (2004).
