

Programação de Computadores I

Prof. Rafael Nunes

O que vamos
aprender neste Curso?

O que vamos aprender neste Curso?

- Aprender os *conceitos básicos* da linguagem de programação C...
- Ela tem se tornado cada dia mais popular, devido à sua *versatilidade* e ao seu *poder*.
- Uma das grandes vantagens do C é que ele possui tanto características de *"alto nível"* quanto de *"baixo nível"*.

Pré-requisitos para o Curso

Pré-requisitos para o Curso

- Apesar de ser bom, não é pré-requisito do curso um conhecimento anterior de *linguagens de programação*.
- É importante uma familiaridade com *computadores*. O que é importante é que você tenha vontade de aprender, *dedicação* ao curso
- E que acompanhe atentamente as discussões que ocorrem na sala de aula do curso.

Quando e como surgiu a
Linguagem C?

Histórico



- O C nasceu na década de 70.
- Seu inventor, *Dennis Ritchie*, implementou-o pela primeira vez usando um *DEC PDP-11* rodando o sistema operacional *UNIX*.

Histórico

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



Onde e de que maneira
podemos utilizar a Linguagem C?

Onde posso utilizar?

- O C é uma linguagem de programação genérica que é utilizada para a *criação de programas* diversos:
 - processadores de texto (Vi, Vim);
 - planilhas eletrônicas;
 - sistemas operacionais (UNIX, Minix, Linux);
 - programas de comunicação;
 - e no nosso caso GAMES!
 - http://cplus.about.com/od/cgames/C_Games_with_Source_Code.htm
 - http://en.wikipedia.org/wiki/List_of_open-source_video_games

Onde posso utilizar?

- programas para a automação industrial;
- gerenciadores de bancos de dados (MySQL);
- programas de projeto assistido por computador;
- programas para a solução de problemas da Engenharia (Matlab);
- Física, Química e outras Ciências, etc. ...

Uma prévia...

ANSI C (Linux x Windows)

- Estudaremos a estrutura do *ANSI C*, o C padronizado pela ANSI.
- Veremos ainda algumas funções comuns em compiladores para alguns sistemas operacionais (*Windows \ Linux*).
- Quando não houver equivalentes para as funções em outros sistemas, apresentaremos *formas alternativas* de uso dos comandos.

A Linguagem C

Parte 1

Já sabemos que o C é...

Case Sensitive

Letras Maiúsculas e Minúsculas
fazem diferença!

Case Sensitive

- *Maiúsculas* e *Minúsculas* fazem diferença
- Se declararmos uma variável com o nome resultado ela será diferente de:
 - Resultado
 - RESULTADO
 - ReSuLtAdO
 - e rEsUITaDo
- Da mesma maneira, os comandos do C *if* e *for*, por exemplo, só podem ser escritos em minúsculas
 - O compilador não irá interpretá-los como sendo comandos, mas sim como variáveis.

Entendendo *a estrutura* dos programas em C

Entendendo a estrutura dos programas em C

- Basicamente qualquer programa estruturado está organizado da seguinte maneira:
 1. **Tudo que eu vou utilizar**
 - Funções Externas ao seu programa
 - Comandos para o pré-processador
 - Variáveis Globais
 2. **As Funções** ou apenas os **protótipos das funções** que serão implementadas por vocês
 3. **A Função Principal**
 - `main ()`
 4. **As Funções** do seu programa (Se vc apenas declarou os protótipos no item 2)

Analizando meu *Primeiro Programa*

Progama1 - Alomundo

- 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
Alomundo.c***

Os 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 - Alomundo.c

Comentário de Linha

- Abrange apenas uma linha
- Para escrever um comentário de linha:
- Digite duas vezes a barra p/ direita “/” antes do comentário → “//”
- Outro ex:

```
01 //Este é um comentário de linha
02 //Função Principal
03 int main ( ) {
04     ...
05 } //Fim
```


Comentário de Bloco

Linha 01 - Alomundo.c

Comentário de Bloco

- Abrange todo um bloco e não apenas uma linha
- Para escrever um bloco de comentário:
 - Inicie o bloco com uma barra p/ direita “/” seguida por um asterisco “*”
 - Termine o bloco com o inverso, um “*” seguido por uma barra para a direita “/”

- Outro ex:

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

Diretivas de Compilação

Linha 02 - Alomundo.c

Diretivas de Compilação

- Uma *diretiva de compilação* é uma instrução para o compilador.
- Inicia-se o primeiro caractere da linha com um “#”
- Algumas diretivas de compilação:
 - include: inclui o arquivo indicado
 - define: define uma macro
 - if, elif, else, endif: compilação condicional

Diretivas 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;
- Toda vez que você quiser usar uma destas funções deve-se incluir este comando.
- O C possui *diversos* arquivos-cabeçalhos (*Bibliotecas*).

Função Principal

Linha 03 - 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()

Professor, o que significa aquele *int*
imediatamente antes do *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 sobre tipos de dados, e também sobre funções, em outra aula

E as chaves “{ }”
após o main?

Linhas 04 e 10 - Alomundo.c

As Chaves “{ }”

- Na linha 4 vemos o caractere *abre-chaves*, um delimitador de bloco “{”
 - Cumpre a mesma função que *begin* em Pascal, serve para abrir um bloco de código.
- Na linha 10 vemos o caractere *fecha-chaves*, outro delimitador de bloco “}”
 - Cumpre a mesma função do *end* em Pascal, serve para fechar um bloco de código
- Neste caso, o que estiver dentro do *par de chaves* pertence a função *main()*

E a função printf() ?

Linha 6

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() ?

O “\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.

Exercícios

Encontre o *Erro*, se houver!

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

```
/* hello.c - Imprime a mensagem "Alo!" na  
tela
```

```
Int Main()
```

```
{
```

```
    printf("Alo!\n");
```

```
    return 0;
```

```
}
```

Encontre o *Erro*, se houver!

```
#include <stdio.h>
/* hello.c - Imprime a mensagem "Alo!" na
tela */
int main()
{
    printf("Alo!" \n);
    return 0;
}
```

Encontre o *Erro*, se houver!

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

```
int main(){printf("Alo!\n");return 0;}
```

Encontre o *Erro*, se houver!

```
#include <stdio.h>
/* hello.c - Imprime a mensagem "Alo!" na
tela */
int main()
{
    printf(Alo!\n);
    return 0;
}
```

Responda

Porque a função `main()` deve conter o comando `return`?

Qual a importância da função `main()`?