Linguagem de Programação I

- Introdução à Linguagem C -

Prof. Ulysses Santos Sousa

ulyssessousa@ifma.edu.br

Aula 01

Roteiro

- **Histórico da Linguagem C**
- **Compiladores x Interpretadores**
- **Estrutura de um programa C**
- **Função *main()***
- **Função *printf()***
- **Função *system()***
- **Pré-processador e diretiva #*include***
- **Variável (tipos e modificadores)**

Por que estudar C?

- C é uma linguagem que utiliza conceitos fundamentais nos quais outros conceitos se baseiam
- As linguagens C++ e C# surgiram a partir da linguagem C. A linguagem Java também possui a sintaxe semelhante a da linguagem C.
- Programas que não precisam do paradigma de Programação Orientada a Objetos podem ser escritos em C.

Fonte:
https://www.streetdirectory.com/travel_guide/114363/programming/10_reasons___why_c_should_be_your_first_programming_language.html

Por que estudar C?

- Programas escritos em C, tem um desempenho (velocidade de execução) bem superior a programas escritos em outras linguagens.
- Boa parte dos sistemas operacionais Window, Linux e Unix ainda são escritos em C.
- Os drivers de novos dispositivos são escritos em C, pois C proporciona ao programador acessos a elementos básicos do computador como: acesso direto à memória através de ponteiros.

Fonte:
https://www.streetdirectory.com/travel_guide/114363/programming/10_reasons___why_c_should_be_your_first_programming_language.html

Por que estudar C?

- C é uma linguagem ideal para programação de sistemas embarcados, que precisam de execução rápida com pouca memória.
- Jogos 3D usam C em seus núcleos. Razão: resposta rápida e imediata aos comandos.
- C é uma linguagem que combina a melhor parte das linguagens de alto nível com as linguagens de baixo nível.

Fonte:
https://www.streetdirectory.com/travel_guide/114363/programming/10_reasons___why_c_should_be_your_first_programming_language.html

Histórico da Linguagem C

- Criada nos laboratórios Bell por Dennis Ritchie na década de 70;
- Foi originada a partir de uma outra chamada BCPL;
- Devido à popularidade da linguagem, formou-se um comitê para a padronização da linguagem;
- Ficando conhecido como C ANSI.

Compiladores x Interpretadores

- **Interpretadores**
  - Lê cada instrução e, constatando a consistência de sua sintaxe, converte para linguagem de máquina e a executa.
  - Precisa estar presente todas as vezes que vamos executar o programa e o trabalho de checagem de sintaxe e tradução deverá ser repetido.

Compiladores x Interpretadores

- **Compiladores**
  - Lê cada instrução, convertendo para linguagem de máquina, mas não executa imediatamente.
  - Cria um arquivo .OBJ que, a ele, serão agregado outras rotinas da linguagem.
  - Rotinas essas feitas por um "linkeditor", que também cria o .EXE
  - Desnecessária a presença do compilador para executar o .EXE.

Fases para um programa em C ser executado

- Edição
- Pré-processamento
- Compilação
- Linking (ligação)
- Carregamento
- Execução

Fases para um programa em C ser executado

- **Edição**
  - Digitação e correção de erros no arquivo .c
- **Pré-processamento**
  - Utiliza comandos especiais chamados de diretivas de *pré-compilação.*
  - Inclui outros arquivos no arquivo a ser compilado e substitui símbolos especiais por textos de programas.
- **Compilação**
  - Traduz o programa em C para o código em linguagem de máquina (código objeto).

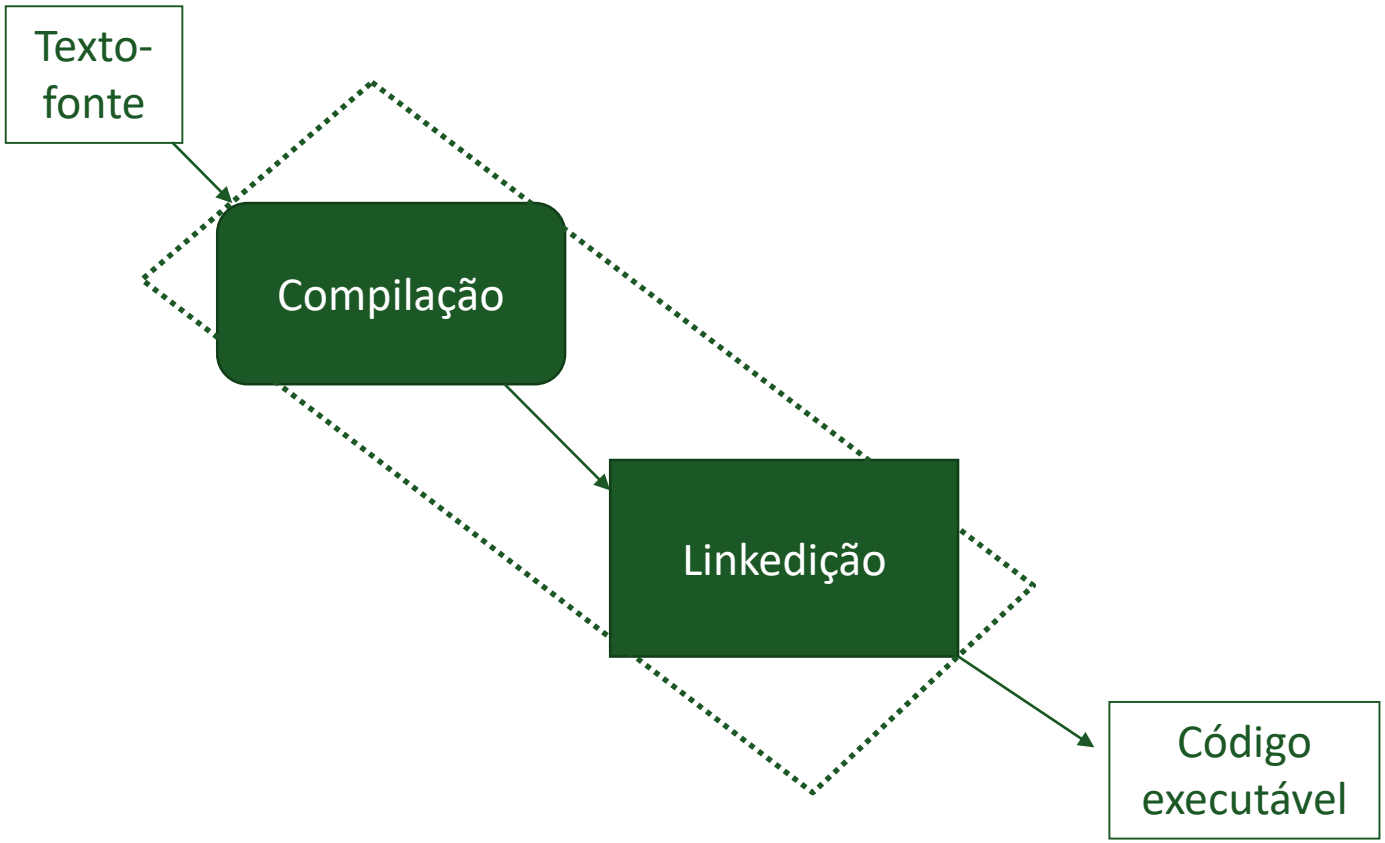
Fases para um programa em C ser executado

- **Linking (ligação)**
  - O Linkeditor une funções compiladas separadamente em um programa. Ele combina as funções da biblioteca padrão C com o código escrito pelo programador.
  - Essa fase gera um programa executável.
- **Carregamento**
  - Um programa deve ser colocado na memória antes que possa ser executado pela primeira vez.
  - O *carregador (rotina de carga* ou *loader)* apanha a imagem executável do disco e transfere para a memória.
- **Execução**
  - O computador, sob o controle de sua CPU, executa as instruções do programa, uma após outra.

Arquivos de um programa em C

- **Programa Fonte**
  - São códigos escritos com o auxílio de um processador de textos e são gravados em disco.
  - Esse código possui a extensão .C.
- **Programa Objeto**
  - Arquivo criado durante a fase de compilação.
  - Esse arquivo possui a extensão .OBJ.
- **Executável**
  - Arquivo com extensão .EXE que executa o programa.

Estrutura de Executável



Palavras reservadas

Categoria	Palavras-chave
Tipos de dados	char, int, float, double, void
Modificadores de tipo	long, short, signed, unsigned
Modificadores de tipo de acesso	const, volatile
Classes de armazenamento	auto, extern, static, register
Tipos definidos pelo usuário	struct, enum, union, typedef
Comandos condicionais	if, else, switch, case, default
Comandos de laço	while, for, do
Comandos de desvio	break, goto, return, continue
Operador	sizeof

Estrutura de um Programa C

- **Forma geral**
  - Um programa C consiste em uma ou várias funções.
  - Tais funções podem ou não receber parâmetros e podem ou não retornar valores;
  - A função que **sempre** deve estar presente em um programa C é a função main():

```
tipo nomeFuncao(declaração dos parâmetros){
   declaração de variáveis;
   instrução_2;
   ...
   instrução_n;
   return var_tipo;
}
```

Estrutura de um Programa C

- **Observações**
  - Os parênteses indicam que esta é uma função. Em um programa C, não pode haver duas funções ***main***;
  - O nome da função, os parênteses e as chaves são os únicos elementos obrigatórios de uma função;
  - Todas as instruções devem estar dentro das chaves que iniciam e terminam a função;
  - As instruções C são sempre encerradas por ponto-e-vírgula.

Estrutura de um Programa C

- **Exemplo:**

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

int main(){
    printf("Primeiro programa.");
    system("PAUSE");
    return 0;
}
```

Função *main()*

- Representa o início do programa.
- Todo programa desenvolvido na Linguagem C começa com a função *main()*

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

int main(){
    printf("Primeiro programa.");
    system("PAUSE");
    return 0;
}
```

Função *printf()*

- Não faz parte da definição da linguagem C.
- É uma das funções de entrada/saída presentes na biblioteca padrão, fornecida com os compiladores C.

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

int main(){
    printf("Primeiro programa.");
    system("PAUSE");
    return 0;
}
```

Função *printf()*

- **É utilizada para exibir na tela mensagens e valores processados ou lidos, de acordo com a necessidade do programa.**
- **Sintaxe:**
  - printf(“expr_de_controle”,lista_de_argumentos);
- **A função printf() pode ter um ou vários argumentos.**

Função *printf()*

- A *expr_de_controle* pode conter caracteres que serão exibidos na tela e códigos de formatação que indicam o formato em que os argumentos devem ser impressos.
- Os argumentos devem ser separados por vírgula.
- Exemplo:

```
int main()
{
    …
    printf("Olá pessoal");
    printf("A média da turma eh %f", m);
    …
}
```

Função *system()*

- Executa um comando interno do sistema operacional ou um programa (.exe, .com ou .bat).

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

int main(){
    printf("Primeiro programa.");
    system("PAUSE");
    return 0;
}
```

Diretivas de Compilação

- **Diretivas**
  - São comandos compilados, sendo dirigidos ao pré-processador, que é executado pelo compilador antes da execução propriamente dita.
  - As diretivas de compilação são iniciadas por #.
- **Pré-processador**
  - é um programa que examina o programa fonte em C e executa certas modificações com base em instruções chamadas diretivas.

Diretivas de Compilação

- **Principais diretivas C**
  - #include
  - #define e #undef
  - #ifdef e #endif
  - #ifndef
  - #if
  - #else
  - #elif

Diretivas de Compilação

- **Diretiva *#include***
  - Diz ao compilador para incluir, na hora da compilação, um arquivo específico.
  - Forma geral:
    - #include “nome_do_arquivo”
    - #include <nome_do_arquivo>
  - Usar “” indica o caminho completo de um arquivo, ou se estiver no diretório de trabalho.
  - Usar <> indica uma inclusão dos próprios arquivos do ambiente C, geralmente arquivos .h (header).
  - Não há ponto-e-vírgula depois das diretivas.

Diretivas de Compilação

- **Arquivos mais comuns:**
  - #include <stdio.h>
    - printf(),fclose(),getchar(),gets();
  - #include <ctype.h>
    - Isalpha(),isalnum()
  - #include <conio.h>
    - getch(),getche()

Diretivas de Compilação

- **Diretiva *#define***
  - Forma geral
    - #define nome_da_macro seq_de_caracteres
  - Diz ao compilador que toda vez que encontrar nome_da_macro no programa a ser compilado, ele irá substituí-lo pela seq_de_caracteres fornecida.
  - Exemplo:

```
#define PI 3.14
```

Códigos de formatação para *printf()*

Código	Formato
%c	Caractere simples.
%d	Inteiro decimal com sinal.
%i	Inteiros decimal com sinal.
%e	Notação científica (e minúsculo).
%E	Notação científica (E maiúsculo).
%g	Usa %e ou %f, o que for menor.
%G	Usa %E ou %f, o que for menor.
%o	Inteiro octal sem sinal.
%f	Ponto flutuante
%u	Inteiro decimal sem sinal.
%p	Apresenta um ponteiro
%%	Escreve o símbolo %
%s	String de caracteres
%x	Inteiro hexadecimal

Códigos especiais para *printf()*

Código	Formato
\n	Nova linha
\t	Tabulação
\b	Retrocesso (usado para impressora)
\f	Salto de página de formulário
\a	Beep – Toque do autofalante
\r	CR - Retorno do cursor para o início da linha
\\	\ - Barra invertida
\’	Aspas simples (apóstrofo)
\”	Aspas dupla
\xdd	Representação hexadecimal
\ddd	Representação octal

Variável

- Espaço de memória reservado para armazenar um certo tipo de dado;
- Possui um nome para referenciar o seu conteúdo;
- Podem conter, a cada tempo, valores diferentes.
- Precisa ser declarada antes de ser utilizada.

Variável

- **Regras necessárias à criação de variáveis:**
  - Só podem ser iniciadas com letras do alfabeto;
  - Não podem conter caracteres especiais, como asteriscos, colchetes, chaves, parênteses etc;
  - Não podem conter espaços em branco. Para palavras compostas pode-se utilizar o *underscore* (_). Ex: logica_de_programacao.
  - Não utilizar acentos e/ou sinais de pontuação.

Variável

- **Tipos de variáveis**
  - O tipo de uma variável informa a quantidade de memória, em bytes que esta irá ocupar
  - Em C existem 5 tipos de variáveis básicas

Tipo	Bytes	Escala
char	1	-128 a 127
int	4	-2.147.483.648 a 2.147.483.647 (ambientes de 32 bits)
float	4	$3,4 \times 10^{-38}$ a $3,4 \times 10^{38}$
double	8	$1,7 \times 10^{-308}$ a $1,7 \times 10^{308}$
void	0	Sem valor

Variável

- **Modificadores de tipo**
  - Todos os tipos básicos podem ser acompanhados com um modificador, exceto o tipo ***void***.
  - Em C há 3 modificadores: ***short, long*** e ***unsigned***.
  - Se o modificador for utilizado sem que haja um tipo especificado para a variável, o compilador assume que o tipo é ***int***.

Variável

- **Modificadores de tipo**

Tipo	Bytes	Escala
short	2	-32.765 a 32.767
long	4	-2.147.483.648 a 2.147.483.647
unsigned char	1	0 a 255
unsigned	4	0 a 4.294.967.294 (ambientes de 32 bits)
unsigned long	4	0 a 4.294.967.294
unsigned short	2	0 a 65.535
long double	10	$3,4 \times 10^{-4932}$ a $3,4 \times 10^{4932}$

Variável

- **Declaração**
  - Na linguagem C, o tipo das variáveis é escrito antes dos identificadores das variáveis.
  - Exemplos:

```
float r1, r2;
int x, y, z;
char a, b, c;
double valor;
```

Termos importantes

- **Código fonte**
  - O texto de um programa que um usuário pode ler, normalmente, interpretado como o programa. O código-fonte é a entrada do compilador C.
- **Código objeto**
  - Tradução do código fonte de um programa em código de máquina que o computador pode ler e executar diretamente. É a entrada para o linkeditor.
- **Linkeditor**
  - Programa que une funções compiladas separadamente em um programa. Ele combina as funções da biblioteca C padrão com o código que você escreveu.
  - A saída do linkeditor é um programa executável.

Termos importantes

- **Biblioteca**
  - Arquivo contendo as funções padrão que seu programa pode usar.
  - Essas funções incluem todas as operações de E/S como também outras rotinas úteis.
- **Tempo de compilação**
  - Os eventos que ocorrem enquanto seu programa está sendo compilado. Uma ocorrência comum em tempo de compilação é um erro de sintaxe.
- **Tempo de execução**
  - Os eventos ocorrem enquanto o seu programa é executado.

Referências

- MIZRAHI, V. V. Treinamento em Linguagem C. 2ª Edição. São Paulo: Person Prentice Hall, 2008.

- SCHILDT, H. C, Completo e Total. 3ª Edição revista e atualizada; Tradução e revisão técnica: Roberto Carlos Mayer. São Paulo: Makron Books, 1996.