Programação de Computadores I
Fundamentos de Programação Estruturada em C
Ciência da Computação

Prof. Daniel Saad Nogueira Nunes

Capítulo 1

Introdução

Capítulo 2

Introdução à linguagem C

Neste capítulo verificaremos a sintaxe básica da linguagem Cbem como seus tipos primitivos de dados e os conceitos preliminares.

2.1 Conceitos preliminares

Começaremos o nosso estudo com um exemplo de programa em C, ilustrado pelo Programa 1.

Programa 1 Olá Mundo em C

Neste programa simples podemos detectar várias coisas interessantes. A linha `##include <stdio.h` especifica que uma biblioteca de entrada e saída padrão do Cdeve ser utilizada, permitindo a impressão de caracteres no console bem como a leitura de valores do teclado. Linhas que iniciam com # são diretivas de pré-processamento. Antes da compilação do programa o pré-processador Cexecuta as ações associadas a cada diretiva. No caso da diretiva `include` , estamos especificando que desejamos usar as funções de determinadas bibliotecas.

A linha `int main(void)` representa a função `main`. Esta função é especial, pois todo o programa Ccomeça a execução desta função. É conhecida como *entry point*

A linha com `printf("Bem vindo ao C!\n ")` faz com que a mensagem `Bem vindo ao C!` seja impressa na saída padrão, o monitor na maior parte das vezes. O commando `printf` é uma função que lida com impressão formatada. Sempre que desejamos imprimir na tela ou em outro dispositivo podemos optar por utilizá-la .

No término do programa, é interessante sinalizar ao Sistema Operacional que tudo ocorreu bem. Neste caso, a convenção é retornar o valor 0 ao fim da função `main`, como disposto na linha `return 0;`

Todas as palavras delimitadas por `/* */` representam comentários. Eles não são executados pelo programa e tem como finalidade documentar o código de modo a deixá-lo mais legível pode desenvolvedores que o utilizarem.

Obervação 1 *Por padrão, a função* `main` *retorna um inteiro, logo deve ser declarada como* `int main(void)` *quando não recebe nenhum parâmetro extra.*

Obervação 2 *É interessante comentar os códigos produzidos, pois ele permite que outra pessoa ou até mesmo o próprio desenvolvedor tenha mais clareza do que está sendo feito.*

Obervação 3 *Normalmente o valor* 0 *é retornado quando o programa termina para identificar que ele terminou sem problemas. Diferentes valores são retornados ao S.O para indicar problemas.*

2.1.1 Compilação

Para executar o programa, precisamos torná-lo primeiramente um executável. O comando `gcc` certifica-se de compilar o código-fonte e ligar os códigos objetos necessários para a produção do executável.

Programa 2 Compilação do código `hello_world.c` para a produção do executável

Este comando gerará o executável `hello_world` a partir do fonte `hello_world.c`.

Obervação 4 *É boa prática de programação utilizar a flag de compilação* `-Wall` *uma vez que ela exibe todos os avisos (warning) de compilação ajudando a produzir programas mais corretos e de acordo com o padrão.*

2.2 Variáveis

Uma variável nada mais é do que um nome para uma posição de memória utilizada para armazenar algum valor. Em `C`, as variáveis possuem um **tipo**, que diz que tipos de valores podem ser armazenados nas posições de memória.

O `C` possui vários tipos primitivos, representados pela Tabela 2.1. Cada tipo primitivo atende uma necessidade básica. O tipo `char` representa caracteres e ocupa um byte, o tipo `int` é utilizado para representação de inteiros. Os tipos `float` e `double` são utilizados para representação de números reais em formato ponto-flutuante, no entanto o `double` possui mais precisão que o `float`, portanto, efetuando cálculos com números reais com maior aproximação.

Tabela 2.1: Tipos primitivos

Tipo	Descrição
char	Armazena um byte, capaz de representar um caractere ASCII
int	um inteiro típico.
float	número ponto-flutuante de precisão simples.
double	número ponto-flutuante de precisão dupla.

O Programa 3 ilustra a declaração de uma variável do tipo `int`. Nele, a variável `i` corresponde a um endereço de memória capaz de armazenar um inteiro.

Programa 3 Declaração de uma variável inteira

```
```

Qualificadores podem ser aplicados a estes tipos básicos, alterando as suas qualidades.

2.2.1 Qualificadores

Os qualificadores `short` e `long` podem ser utilizados para prover diferentes tamanhos de inteiro, portanto, eles podem ser aplicados ao tipo `int`. Normalmente uma variável `short int` ocupará 16 bits, uma variável `int` sem qualificador ocupará 32 bits e uma variável `long int` ocupará ao menos 32 bits.

O qualificador `signed` e `unsigned` permitem dizer se o inteiro ou char utilizado é com sinal ou sem sinal. Ao representar números sem sinal, você consegue representar números maiores, uma vez que o bit utilizado para identificar se o número era positivo ou negativo, agora pode ser utilizado para aumentar a capacidade de representação de números sem sinais. Tipicamente, os tipos inteiros e `char` são `signed`.

A Tabela resume o discutido até então.

2.3 Operadores

Capítulo 3

Estruturas de decisão