

Estrutura de Dados I

2025/2

Nicolas Ramos Carreira

Sumário

1	Intuito	2
2	Fundamentos em C	3
2.1	Sobre a linguagem C	3
2.2	Estrutura de um programa em C	3
2.3	Aspectos da linguagem C	4
2.3.1	Variaveis	4
2.3.2	Tipos de dados	5
2.3.3	Input e output	6
2.3.4	Contantes	7
2.3.5	Operadores	7
2.3.6	Coerção de tipos	7
2.3.7	Condicionais	7
2.3.8	Loops	7
2.3.9	Arrays	7
2.3.10	Struct - Criação de tipos	7
3	Falando sobre ponteiros	8

Capítulo 1

Intuito

O intuito deste documento é documentar o meu aprendizado da disciplina de estrutura de dados 1. Nesta disciplina começamos estudando sobre a linguagem C até entrar nas principais estruturas de dados.

Capítulo 2

Fundamentos em C

2.1 Sobre a linguagem C

A linguagem C é uma das linguagens de programação mais influentes e utilizadas da história da computação. Criada na década de 1970 por Dennis Ritchie nos laboratórios Bell, ela foi projetada para ser uma linguagem de propósito geral, eficiente e próxima do hardware, permitindo alto desempenho.

C é considerada uma linguagem de médio nível, pois combina características de linguagens de baixo nível (como manipulação direta de memória) com recursos de alto nível (como funções e estruturas). Sua sintaxe influenciou muitas outras linguagens modernas, como C++, Java, Csharp e até mesmo Python em alguns aspectos.

É amplamente usada em sistemas operacionais, softwares embarcados, drivers e aplicações que exigem alto desempenho. Além disso, aprender C é um ótimo ponto de partida para entender conceitos fundamentais de programação e arquitetura de computadores.

2.2 Estrutura de um programa em C

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following C code:

```
#include <stdio.h>

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

A imagem acima mostra um programinha extremamente simples em C, um Hello, world. Para iniciar um programa em C, nós sempre começamos declarando a biblioteca principal, que é a `stdio.h` (poderíamos ter outras bibliotecas inclusive, mas essa é a principal e DEVE estar lá).

Depois disso, nós declaramos o local do programa principal, onde fazemos o programa em si.

Um detalhe é que ao final de cada coisa SEMPRE temos que ter o ponto e vírgula (;), pois se não o nosso programa não compila.

2.3 Aspectos da linguagem C

2.3.1 Variáveis

O que são e pra que são usadas

Varível, em linguagens de programação, é basicamente uma posição alocada da memória para guardar uma informação. Variáveis podem ser modificadas pelo programa e devem ser definidas antes de ser utilizadas

Declaração de variáveis em C

Para definir variáveis em C, nós precisamos passar o tipo de dado e nome da variável, no formato: `<tipo de dado> nome-da-variavel`. O tipo de dado deve ser aqueles que são aceitos pela linguagem (inteiro, decimais, caracteres, booleanos..), mas como falaremos sobre tipos de dados mais pra frente, não entraremos em detalhes agora. O nome da variável é algo bem importante a se considerar, pois existem algumas regras e boas práticas importantes quanto a isso:

- Nomes de variáveis devem iniciar com letras ou underscore
- Os caracteres da variável devem ser letras, numeros ou underscore (não utilizar acentos ou símbolos)
- Não utilizar espaço em nomes de variáveis
- Palavras chaves (palavras que são reservadas pela linguagem para fazer determinadas coisas) não podem ser usadas como nomes
- Letras maiúsculas e minúsculas são consideradas diferentes

Só para deixar totalmente claro, as palavras chaves que a linguagem C usa são:

auto	break	case	char	const	continue	do	double
else	for	int	union	static	default	void	return
enum	goto	long	unsigned	struct	extern	while	sizeof
float	if	short	volatile	switch	register	typedef	

Atribuição de valores em variáveis

Tendo o formato <tipo de dado> nome-da-variavel, podemos atribuir valores a elas (ou seja, armazenar valores dentro da memória). Para isso, basta fazer:

<tipo de dados> nome-variavel = valor

2.3.2 Tipos de dados

Como falamos anteriormente na parte de variáveis, quando vamos defini-las, nós temos que declarar o tipo de dado da variável. O tipo de dado define os valores que aquela variável pode assumir e as operações que podem ser realizadas com ela. Os tipos de dados principais são: char, int, float e double

Char

Um byte que armazena

Int

Um inteiro cujo o tamanho do número que pode ser alcançado depende do processador (tipicamente 16 ou 32 bits)

Float

Basicamente números decimais com precisão simples (em C a parte decimal usa ponto e não vírgula)

Double

Também números decimais, mas com precisão dupla. É usados para números muito pequenos (científicos por exemplos) ou muito grandes

Outros tipos

Na imagem abaixo, você poderá ver alguns outros que são utilizados:

Tipo	Bits	Intervalo de valores
char	8	-128 A 127
unsigned char	8	0 A 255
signed char	8	-128 A 127
int	32	-2.147.483.648 A 2.147.483.647
unsigned int	32	0 A 4.294.967.295
signed int	32	-32.768 A 32.767
short int	16	-32.768 A 32.767
unsigned short int	16	0 A 65.535
signed short int	16	-32.768 A 32.767
long int	32	-2.147.483.648 A 2.147.483.647
unsigned long int	32	0 A 4.294.967.295
signed long int	32	-2.147.483.648 A 2.147.483.647
float	32	1,175494E-038 A 3,402823E+038
double	64	2,225074E-308 A 1,797693E+308
long double	96	3,4E-4932 A 3,4E+4932

2.3.3 Input e output

Input e output é basicamente a entrada e a saída de dados. As vezes, podemos querer receber do usuário alguns valores, para fazer alguma coisa com eles e depois entregá-los com modificações. É basicamente isso. Um detalhe é que para o output, não necessariamente nós precisamos ter recebido algo.

Especificadores de formato

Saída com printf()

Vamos começar com a saída de dados. Para exibir algo na tela. Fazemos:

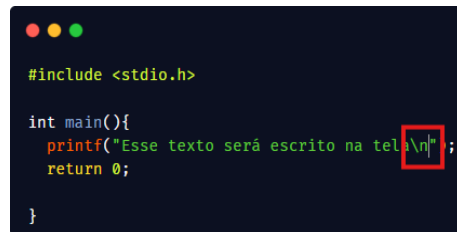
```
#include <stdio.h>

int main(){
    printf("Esse texto será escrito na tela");
    return 0;
}
```

Ao fazer isso, em nosso terminal será exibido o texto que digitamos dentro do printf ("Esse texto será escrito na tela). Veja

Uso do escape no printf()

Um detalhe é que algo que podemos utiliza no printf é caracter de escape . Esse caracter é utilizado sempre ao final do que que queremos escrever na saída e ele serve para quebrar a linha após a saída. Veja:



```
#include <stdio.h>

int main(){
    printf("Esse texto será escrito na tela\n");
    return 0;
}
```

Se fizermos vários printf, por exemplo, e não usarmos o caracter de escape em nenhum deles, o que escrevemos nos prints, ficará tudo junto. Veja

Entrada com scanf()

2.3.4 Constantes

Assim como variaveis, constantes também armazenam um valor na memória do computador. A principal diferença para as variaveis é que esse valor não será alterado. Outra coisa é que para as constantes é obrigatoria a atribuição de valor, diferente das variaveis que podemos simplesmente declará-las sem dar um valor

Declaração de constantes

Para declarar uma constante existem duas formas. Na primeira, devemos utilizar define nome-costante <valor> no começo do programa. Veja:

Outra forma é fazer const <tipo> nome = valor

2.3.5 Operadores

Os operadores são usados para desenvolver diferentes tipos de operações. Com eles podemos fazer operações matematicas, comparativas, logicas e etc. Veremos acerca de cada um dos operadores a seguir

Operadores aritméticos

Operadores relacionais

Operadores lógicos

2.3.6 Coerção de tipos

2.3.7 Condicionais

If-else

Swicth-case

2.3.8 Loops

2.3.9 Arrays

2.3.10 Struct - Criação de tipos

Capítulo 3

Falando sobre ponteiros