estudar.com.vc

Mac em C

Resuminho e Exercícios P3

Resumo Teórico

Vetores e matrizes

Declaramos vetores da seguinte forma:

```
int vetor[50];
```

onde 50 é exemplo de um número inteiro e constante.

Podemos também fazer um #define no início do programa

```
#define TAMANHO 45
int main()
{
   int vetor[50];
   int vetor2[TAMANHO];
}
```

Para inicializar vetores pequenos podemos digitar seu conteúdo:

```
int vetor3[4] = {1, 2, 3, 4};
```

Declaramos matrizes de n dimensões com n colchetes após o nome:

```
int matriz[50][50];
```

Ao declararmos uma função que recebe uma matriz como parâmetro, devemos dizer qual é o tamanho de cada coluna. Adicionalmente, para uma matriz de n dimensões, devemos dizer o tamanho de todas, exceto a que está mais à esquerda. Esta última pode ou não conter o tamanho. Exemplo de declaração de protótipos de funções:

```
int recebe (int matriz[][50]) ;
int recebe (int matriz[50][50]) ;
int recebe (int matriz[][50] [50]) ;
```

Strings e char

Char é o tipo de variável que utilizamos para tratar letras do alfabeto. Porém, em C, char nada mais é que um int de tamanho reduzido (aliás o menor tipo de

variável possível em C), indo de -127 a 127. Podemos imprimir um caractere através do modificador %c no printf. Além disso, se estiver no intervalo mencionado acima, podemos também imprimir a partir de um valor de uma variável de tipo int.

A leitura de um conjunto de caracteres, uma string, é feita com o modificador %s no scanf, e o segundo parâmetro tem que ser um vetor grande o suficiente para conter a string. Além disso, é bom também saber que esse vetor de caracteres sempre terminará com o símbolo '\0', para indicar o término da string.

A biblioteca string.h é muito útil para trabalharmos com strings, contendo vários métodos que poupam bastante trabalho:

strlen(char *str) -> retorna o tamanho da string

strcpy(char *dest, const char *src) -> copia a string src para a string dest

strcat(char *dest, const char *src) -> coloca a string src no final da dest.

Outras funções úteis podem ser vistas pesquisando o nome da biblioteca na internet.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Funções Recursivas

São funções que em algum momento dentro de seus códigos chamam ela mesmas. São muito utilizadas para quebra de problemas na força bruta, onde reduz-se o problema a um ou mais casos fáceis recorrentes e depois chama-se a própria função com parâmetros diferentes. Esses casos chamamos de base da recursão. É fundamental que existam esses casos, nos quais a função retorna, necessariamente (independentemente de ser void ou não).

Inclusão de outros arquivos-código

Também é muito comum que separemos nosso código em vários arquivos diferentes, quando temos projetos muito grandes. Isso facilita a organização e a compreensão do projeto. Para incluir um arquivo que contenha funções úteis de impressão de matrizes, por exemplo, basta fazer um include, assim:

#include “impressaoMatrizes.c”

Isso supõe que ambos arquivos estejam no mesmo diretório.

Dicas

vetor[i++] = j; -> atribui j pra posição e soma 1 no i

vetor[++i] = j; ->Soma 1 no i e atribui j pra posição resultante

Escrever (int) na frente do nome de uma variável faz com que o compilador leia a variável com int, mesmo que tenha sido declarada como double por exemplo. Nesse caso em específico, *(**int**)* ***a*** para um ***a*** = 4.36 vai resultar em 4. Ou seja, equivale a truncar o número. Essa prática é chamada de casting.

Exercícios:

1. Exercício de Prova

P2 2016, questão 1

Faça um programa em C que recebe um inteiro $n > 0$ e uma sequencia de n valores inteiros, imprimindo "ALTERNANTE" se a sequência é *alternante crescente-decrescente* (ACD) e, em caso contrário, imprime "NÃO". A sequencia é alternante crescente-decrescente se alternar números menores e maiores como abaixo.

Exemplos de ACD: {1}, {-1, 1, -1, 1}, {-1, 1, -1, 1, -1, 1}, {-2, 0, -1, 1}

Exemplos de NAO ACD: {1, -1, 1, -1}, {-1, -1, 1, -1, 1}, {-2, 0, 0, -1, 1}

2. Exercício de Prova

P2 2016, questão 2

Um anagrama (do grego ana = "voltar" ou "repetir" + graphein = "escrever") é uma espécie de jogo de palavras, resultando do rearranjo das letras de uma palavra ou frase para produzir outras palavras, utilizando todas as letras originais exatamente uma vez. Desconsiderando acentos, um exemplo conhecido é o nome da personagem Iracema, claro anagrama de América, no romance de José de Alencar.

Faça uma função que testa se duas palavras fornecidas em *strings* são anagramas, i.e., elas devem conter o mesmo número de ocorrências dos seus caracteres. O usuário deve digitar as duas palavras e seu programa deve imprimir ANAGRAMAS se o forem, ou NÃO em caso contrário.

ATENÇÃO: Deve-se ignorar diferenças entre maiúsculas e minúsculas, i.e., 'a' e 'A' devem ser consideradas iguais.

3. Exercício de Prova

P2 2016, questão 3

Dado um inteiro *n > 0* e uma sequência com n valores inteiros, determinar o conjunto de representantes (CR) da sequência e imprimir a frequência de cada um deles. O CR será um vetor com elementos inseridos na ordem de aparição, ou seja, se a primeira entrada do valor 2 ocorreu antes da primeira entrada de -1, então 2 estará à esquerda de -1 em CR (índice do 2 menor que o do -1). Seu programa deverá, ao final, imprimir cada elemento de CR e quantas vezes este valor apareceu. A ordem de impressão deverá ser a ordem usual crescente de índice (ou seja, 0, 1, e assim por diante). Fazer obrigatoriamente com duas funções, as funções *conjunto* e *imprime*, ambas com três parâmetros, a primeira do tipo int devolvendo o número de elementos distintos na sequência (tamanho de CR) e a segunda vazia (void) deverá fazer a impressão dos elementos e sua frequência.

4. Exercício de Prova

P2 2016, questão 4

Faça uma função que leia *n* e gera uma matriz n x n com os números 1 até n^2, seguindo um padrão crescente de números consecutivos dispostos em forma de espiral, conforme os exemplos.

Exemplos: n = 1 imprime $\begin{vmatrix}1\end{vmatrix}$, n = 2 imprime $\begin{vmatrix}1 & 2\\ 4 & 3\end{vmatrix}$, n = 3 imprime $\begin{vmatrix}1 & 2 & 3\\ 8 & 9 & 4\\ 7 & 6 & 5\end{vmatrix}$

Gabarito em arquivos separados