Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Engenharia Elétrica

CSI488 – ALGORITMOS ESTRUTURAS DE DADOS I (TURMA 22)

ALOCAÇÃO DINÂMICA

Jhonatan Figueiredo Almeida

João Monlevade
Março de 2022

CSI488 – ALGORITMOS ESTRUTURAS DE DADOS I (TURMA 22)

ALOCAÇÃO DINÂMICA DE MEMÓRIA ATIVIDADE EXTRA

Discente: Jhonatan Figueiredo Almeida Matrícula: 20.1.8164
Professor: Alexandre Magno de Souza

João Monlevade
Março de 2022

Exercício: Página c620.html

Enunciado:

(a) -Explique a diferença entre
p++; (*p)++; *(p++);
-O que quer dizer *(p+10);?
-Explique o que você entendeu da comparação entre ponteiros.

Solução:

- p++: incrementa o ponteiro, ou seja o endereço. Após esta instrução, o ponteiro p passará a apontar para a posição de memória imediatamente superior. Se em um vetor, o ponteiro passará a apontar a próxima posição do vetor.
- (*p)++: Incrementa o conteúdo apontado por p, ou seja, o valor armazenado na variável para qual p está apontando.
- *(p++): Incrementa p (como em p++) e acessa o valor encontrado na nova posição. Se em um vetor, esta expressão acessa o valor da posição imediatamente superior a armazenada em p antes do incremento.
- *(p+10) Acessa o valor encontrado 10 posições a frente de p. Neste caso, o apontador não é incrementado. Se em um vetor, irá acessar a décima posição após a que está sendo apontada.
- Dois ponteiros, como outras variaveis, podem ser comparados. Podemos verificar por exemplo se dois ponteiros apontam para a mesma posição de memória verificando se p1 == p2 ou se p1 != p2
Podemos comparar se um ponteiro e 'menor' ou 'maior' que outro, ou melhor, se aponta para uma posição superior a de outro.

b) Enunciado

Qual o valor de y no final do programa? Tente primeiro descobrir e depois verifique no computador o resultado. A seguir, escreva um /* comentário */ em cada comando de atribuição explicando o que ele faz e o valor da variável à esquerda do '=' após sua execução.

```
int main()
{
int y, *p, x;
```

```
y = 0;
p = &y;
x = *p;
x = 4;
(*p)++;
x--;
(*p) += x;
printf ("y = %d\n", y);
return(0);
}
```

Solução:

O valor de y é 4, e o programa comentado fica:

```
int main()
{
int y, *p, x;
y = 0; /* atribui o valor 0 a y => y=0 */
p = &y; /* atribui o endereco de y ao ponteiro p
p contem o endereco de y (ex:DS:FFF4)*/
x = *p; /* atribui o conteudo de onde p aponta
(valor de y) para x, que passa a valer 0 */
x = 4; /* atribui 4 a x */
(*p)++; /* incrementa de 1 o conteudo de onde p aponta,
alterando o valor de y para 1 */
x--; /* decrementa 1 de x => x = 3 */
(*p) += x; /* adiciona x ao conteudo de onde p aponta,
alterando o valor de y para 4 */
printf ("y = %d\n", y); /* imprime "y = 4" */
}
```

Exercício: Página c630.html

Enunciado:

Fizemos a função StrCpy(). Faça uma função StrLen() e StrCat() que funcionem como as funções strlen() e strcat() de string.h, respectivamente.

Solução:

Função StrLen

```
#include <stdio.h>
StrLen (char *str)
{
int tamanho = 0;
while (*str)
{
tamanho++;
str++;
}
return tamanho;
}
```

Função StrCat

```
#include <stdio.h>
#include <string.h>
StrCat (char *primeira, char *segunda)
{
char *p;

/* --->p aponta para o final da primeira string */
p = primeira+strlen(primeira);
while (*segunda)
{
*p = *segunda;
p++;
```

```
segunda++;
}
*p = '\0';
}
```

Função strend

```
#include <stdio.h>
#include <string.h>
strend (char *str, char *t)
{
char *p;
p = str + strlen(str) - strlen(t); /* p aponta para o final da string str -
tamanho de t */
while (*t)
{
if (*p != *t) return 0;
p++;
t++;
}
return 1;
}
```

Exercício: Página c650.html

Verifique o programa abaixo. Encontre o seu erro e corrija-o para que escreva o numero 10 na tela.

```
#include <stdio.h>
int main()
{
int x, *p, **q;
p = &x;
q = &p;
x = 10;
printf("\n%d\n", &q);
```

return(0);
}

Solução:
O programa contém um erro na linha do printf, onde ele manda imprimir o endereço de q (&q). Na realidade, para se imprimir o valor 10 (valor de x) deve-se imprimir o valor apontado pelo valor apontado por q. Veja o esquema:
x = 10;
p aponta para x;

q aponta para p;

==> *q é igual a p ; como *p é igual a x, basta escrever *(*q) para se ter x.
Logo, o printf ficaria:
printf("\n%d\n", **q);

Exercício : Página c660.html
Enunciado:

Escreva um programa que declare uma matriz 100x100 de inteiros. Você deve inicializar a matriz com zeros usando ponteiros. Preencha depois a matriz com os números de 1 a 10.000 usando ponteiros.

Solução:
```
/* Problema das matrizes ---------- */
#include <stdio.h>
main ()
{
int mat[100][100];
int *p;
int i, j, soma = 0;
p = &mat[0][0]; /* Inicializa o ponteiro no inicio da matriz */
/* Inicializando a matriz com zeros.. */
for (i=0; i<100; i++)
```

```
for (j=0; j<100; j++)
{
*p = 0;
p++;
}
/* Preenchendo a matriz com numeros */
p = &mat[0][0];
for (i=0; i<100; i++)
for (j=0; j<100; j++)
{
*p = soma;
soma++;
p++;
}

Return 0;
}
```

Escreva um programa que declare uma matriz 100x100 de inteiros. Você deve inicializar a matriz com zeros usando ponteiros para endereçar seus elementos. Preencha depois a matriz com os números de 1 a 10000, também usando ponteiros.

6.7 Avaliação

1- Seja um vetor declarado por

```
int vet[10];
```

Qual elemento deste vetor é acessado quando se escreve vet[2] ?

a. Primeiro elemento

b. Segundo elemento

● Terceiro elemento x

d. Quarto elemento

e. Nenhuma das opções anteriores

2- Se declararmos um vetor como:

```
int vet[30]
```

a instrução abaixo acessa corretamente os elementos deste vetor?

```
for (j=0; j <= 30; j++)
   vet[j] = j*j;
```

a. Sim

● Não ✓

3- Seja a matriz matrx declarada e inicializada por:

```
int matrx[][4] = {1,2,3,4,5,6,7,8,9,10,11,12};
```

O que conterá o elemento matrx[1][2] ?

a. 2

b. 5

c. 6

e. Nenhuma das opções anteriores

4- Se uma string for declarada como:

```
char str[20];
```

o número máximo de caracteres que poderão ser lidos e armazenados nela é:

a. 18

b. 19

c. 20

d. 21

5- Qual função pode ser usada para determinar o comprimento de uma string?

a. gets

b. strcpy

c. strcat

d. strlen

e. strcmp

6- Qual das instruções abaixo é correta para declarar um ponteiro para inteiro?

a. *int pti;

b. *pti;

c. &i;

d. int_pti pti;

e. int *pti;

7- Seja a seguinte seqüência de instruções em um programa C:

```
int *pti;
int i = 10;
pti = &i;
```

Qual afirmativa é **falsa**?

a. pti armazena o endereço de i

b. *pti é igual a 10

c. ao se executar *pti = 20; i passará a ter o valor 20

d. ao se alterar o valor de i, *pti será modificado

● pti é igual a 10

8- Se i e j são variáveis inteiras e pi e pj são ponteiros para inteiro, qual atribuição **é ilegal**?

a. pi = &i;

● *pj = &j;

c. pj = &*&j;

d. i = *&*&j;

e. i = (*pi)+++*pj;

9- Seja a seguinte seqüência de instruções em um programa C:

```
int *pti;
int veti[]={10,7,2,6,3};
pti = veti;
```

Qual afirmativa é **falsa**?

a. *pti é igual a 10

b. *(pti+2) é igual a 2

c. pti[4] é igual a 3

● pti[1] é igual a 10

e. *(veti+3) é igual a 6

10- Na seqüência de instruções abaixo:

```
float f;
float *pf;
pf = &f;
scanf("%f", pf);
```

a. Efetuamos a leitura de f

b. Não efetuamos a leitura de f

c. Temos um erro de sintaxe

d. Deveríamos estar usando &pf no scanf

e. Nenhuma das opções anteriores

11- Seja a seguinte seqüência de instruções

```
int i=10, j=20;
int *pti, *ptj;
pti = &i;
ptj = &j;
```

Qual expressão **não é válida**?

a. j = pti == ptj;.

b. i = pti-ptj;

c. pti += ptj;

d. pti++;

e. i = pti || ptj;

12- Seja a declaração:

```
int matr[][4] = {1,2,3,4,5,6,7,8,9,10,11,12}
```

Qual afirmativa é falsa?

a. **matr é igual a 1

b. *(*(matr+1)+2) é igual a 7

c. *(matr[2]+3) é igual a 12

d. (*(matr+2))[2] é igual a 11

e. *((*matr)+1) é igual a 5