ESCOLA PROFISSIONAL
CRISTÓVÃO COLOMBO

www.epcc.pt

MADEIRA 14-20 · REGIÃO AUTÓNOMA DA MADEIRA · PORTUGAL 2020 · UNIÃO EUROPEIA · Fundos Europeus Estruturais e de Investimento

Sistemas Operativos – Processamento Computacional

Arrays

Autor da apresentação

09/09/2014

Arrays

Um *array* é um conjunto de variáveis do mesmo tipo que podem ser referenciadas por um identificador comum. Em C um *array* consiste num conjunto de posições de memória contíguas, o menor endereço corresponde ao primeiro elemento e o maior endereço corresponde ao último elemento. Um *array* pode ter apenas uma dimensão (*array* unidimensional ou vector) ou várias (*array* multidimensional ou matriz).

Arrays - Sintaxe

Tipo nome_array[dimensão];

O índice permite referenciar individualmente cada um dos elementos dum *array* inicia-se em 0 (zero).

Referencia-se com parênteses rectos ‘[‘ ‘]’.
Exemplo :

_Array[2]= 12;

Arrays - Exemplo

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

#define DIM 13
int main(){
    int arr[DIM];
    //int i;
    for (int lin=0; lin<DIM; lin++)
        arr[lin]=lin;

    for (int lin=0; lin<DIM; lin++)
        printf("Array[%d]=%d\n",lin,arr[lin]);
}
```

Arrays – Exemplo 2 Leitura via terminal

```
int main(){
 int z, vet[4];
 for(z=0; z<4; z++){
    printf("Entre com  o elemento matriz[%d]: ",z+1);
     scanf("%d", &vet[z]);
 }

 for(z=0; z<4; z++)
    printf("%d\t",vet[z]);

 printf("\n");
 system("pause");
 return(0);
}
```

Arrays – Exemplo 3 Leitura via terminal

```c
#define DIM 4

double media(int arr[],int size){
    int i;
    double avg,sum=0;
    for (int lin=0; lin<DIM; lin++)
        sum += arr[lin];
    avg = sum /size;
    return avg;
}
int main(){
 int vet[DIM] = {123,34,67,12};
 double avg;
/* Não existe método nativo temos que o calcular
   Tamanho do array em bytes / pelo tamanho do tipo em bytes
   Exemplo Vetor tem 4 inteiros(4 bytes cada) <=>
   4*4 = 16 bytes / 4 bytes(do tipo int) <=> 4 elementos
*/
 int sizeVet = sizeof(vet)/sizeof(int);
 avg = media(vet,sizeVet);
 printf("A Média de %d valores é : %f",sizeVet,avg);
}
```

OU

```c
double media(int *arr,int size){
```

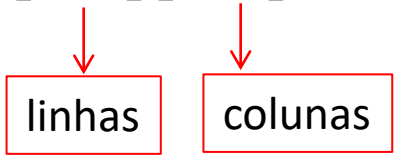
Arrays multidimensionais

A linguagem C permite *array*'s multidimensionais. Para declarar um *array* bidimensional de inteiros.
Sintaxe:

tipo nome_array[dimensão][dimensão];

Exemplo:

int tab[10][20];



Arrays multidimensionais - Exemplo

```
#define DIML 3
#define DIMC 4
void imprimeMatriz(int mat[][DIMC]){
    int lin,col;
    //Imprimindo a matriz original
    printf("Matriz original\n");
    for (lin=0;lin<DIML;lin++){
        for (col=0;col<DIMC;col++)
            printf("%d\t",mat[lin][col]);
        printf("\n\n");
    }
}
int main(){
int a[DIML][DIMC] = {
   {0, 1, 2, 3} ,   /*  initializers for row indexed by 0 */
   {4, 5, 6, 7} ,   /*  initializers for row indexed by 1 */
   {8, 9, 10, 11}   /*  initializers for row indexed by 2 */
};
int b[DIML][DIMC] = {0,1,2,3,4,5,6,7,8,9,10,11};
imprimeMatriz(a);
imprimeMatriz(b);
}
```

Arrays multidimensionais – Exemplo 2

```c
#define DIML 3
#define DIMC 4
void imprimeMatriz(int mat[][DIMC]){
    int lin,col;
    //Imprimindo a matriz original
    printf("Matriz original\n");
    for (lin=0;lin<DIML;lin++){
        for (col=0;col<DIMC;col++)
            printf("%d\t",mat[lin][col]);
        printf("\n\n");
    }
}

void preencheMatriz(int mat[][DIMC]){
    int lin,col, tab;
    for (lin=0; lin<DIML; lin++)
        for (col=0; col<DIMC;col++){
            printf("Digite ELEMENTO da linha %d, coluna %d da matriz: ",lin+1,col+1);
            // aqui no scanf preenchemos a matriz
            scanf("%d", &mat[lin][col]);
        }
}
```

MAIN

```c
int main(){
    int a[DIML][DIMC];
    preencheMatriz(a);
    imprimeMatriz(a);
}
```

Chamadas recursivas– Exemplo

```c
#include <stdio.h>
#include <stdlib.h>
#define DIM 10
/*
Cada vez que uma chamada recursiva a programa() é feita, a instrução
printf() que vem em seguida fica pendente, sendo executada somente
quando o controle retorna das chamadas recursivas. Instruções pendentes
são mantidas numa estrutura de pilha, de modo que a última delas
é a primeira a ser executada. Isso faz com que os valores sejam exibidos
na ordem inversa àquela em foram recebidos pela função prog()*/
void programa(int n){
    if(n==0) return;
    programa(n-1);
    printf("%d ",n);
}

int main(){
    programa(10);
}
```