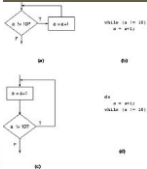


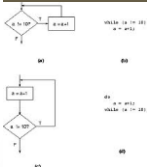
Estruturas de Dados

- Prof. Edkallenn Lima
- edkallenn@yahoo.com.br (somente para dúvidas)
- Blogs:
 - <http://professored.wordpress.com> (Computador de Papel – O conteúdo da forma)
 - <http://professored.tumblr.com/> (Pensamentos Incompletos)
 - <http://umcientistaporquinzena.tumblr.com/> (Um cientista por quinzena)
 - <http://eulinoslivros.tumblr.com/> (Eu Li nos Livros)
 - <http://linabiblia.tumblr.com/> (Eu Li na Bíblia)
- Redes Sociais:
 - <http://www.facebook.com/edkallenn>
 - <http://twitter.com/edkallenn>
 - <https://plus.google.com/u/0/113248995006035389558/posts>
 - [Pinterest: https://www.pinterest.com/edkallenn/](https://www.pinterest.com/edkallenn/)
 - [Instagram: http://instagram.com/edkallenn](https://instagram.com/edkallenn) ou [@edkallenn](https://instagram.com/edkallenn)
 - [LinkedIn: br.linkedin.com/in/Edkallenn](https://br.linkedin.com/in/Edkallenn)
 - [Foursquare: https://pt.foursquare.com/edkallenn](https://pt.foursquare.com/edkallenn)
- Telefones:
 - 68 8401-2103 (VIVO) e 68 3212-1211.
- Os exercícios devem ser enviados SEMPRE para o e-mail: edkevan@gmail.com ou para o e-mail: edkallenn.lima@uninorteac.edu.br



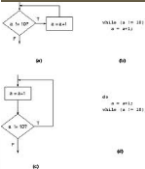
Agenda

- Matrizes bidimensionais
- Inicialização de matrizes de duas dimensões
- Matrizes como argumentos
- Matrizes de strings
- Matrizes multidimensionais



Matrizes (Arrays) Bi e multidimensionais

- Os arrays em C podem ter **vários subscritos**.
- São chamadas matrizes **multidimensionais**
- O uso mais comum é para representar **tabelas** de **valores** que consistem em informações organizadas em linhas e colunas.
- Para **identificar** um elemento **específico** de uma tabela devemos especificar **dois subscritos**: o **primeiro** (por convenção) indica a **linha** do elemento, e o **segundo** (por convenção) indica a **coluna** do elemento.
- Os arrays com dois subscritos são chamados arrays bidimensionais



Matrizes multidimensionais

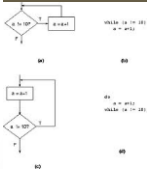
- O padrão ANSI declara que um sistema ANSI C deve suportar, pelo menos, **12 subscritos** de array
- Array **a** com **dois** subscritos (bidimensional):

	Coluna 0	Coluna 1	Coluna 2	Coluna 3
Linha 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Linha 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Linha 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Nome do array

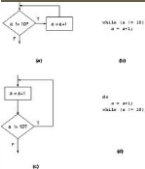
Subscrito da linha

Subscrito da coluna



Matrizes multidimensionais

- O array(matriz) anterior contém 3 linhas e 4 colunas, portanto diz-se que é uma **matriz 3-por-4**.
- Em geral uma matriz com m linhas e n colunas é uma **matriz m-por-n**
- Cada elemento de **a** é identificado por um nome da forma **a[i][j]**: **a** é o nome, **i** é o subscrito da linha e **j** é o subscrito da coluna
- Observe que mantém-se o subscrito inicial igual a 0.



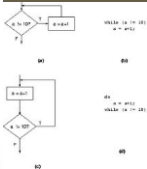
Arrays bidimensionais

- Exemplo: Criar uma matriz que tenha 100 linhas por 50 colunas

```
int mat[100][50];
```

```
mat[0][1] = 99;
```

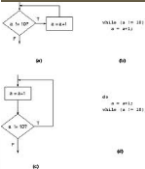
	0	1	...	49
0		99		
1				
...				
mat[0][1]				
99				



Arrays bidimensionais

- Cada elemento da matriz tem **todas** as **características** de uma **variável** e pode aparecer em **expressões** e atribuições (respeitando os seus tipos)

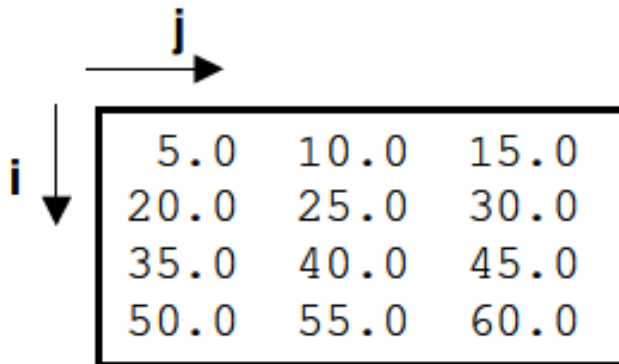
```
mat[0][1] = x + mat[1][5];  
if (mat[5][7] > 0)
```



Matrizes bidimensionais

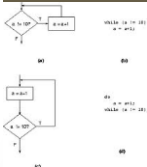
- Matriz bidimensional:

```
float m[4][3] = {{ 5.0, 10.0, 15.0},
                 {20.0, 25.0, 30.0},
                 {35.0, 40.0, 45.0},
                 {50.0, 55.0, 60.0}};
```



	j →		
i ↓	5.0	10.0	15.0
	20.0	25.0	30.0
	35.0	40.0	45.0
	50.0	55.0	60.0

60.0	
55.0	
50.0	
45.0	
40.0	
35.0	
30.0	
25.0	
20.0	
15.0	
10.0	
5.0	
152	104

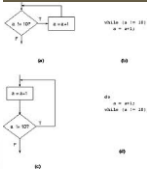


Leitura e exibição de matrizes bidimensionais (sem funções – AINDA)

```

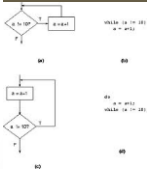
1  #include <stdio.h>
2  #include <stdlib.h>
3  /* Função : Exemplos de matrizes
4  Autor : Edkallenn - Data : 10/04/2012 */
5  #define LINHA 4
6  #define COLUNA 3
7  main() {
8      int i, j, b[LINHA][COLUNA];
9      //lê
10     for(i=0;i<LINHA;i++){           //linha
11         for(j=0;j<COLUNA;j++){       //coluna
12             printf("Digite o elemento b[%d][%d]:", i+1, j+1);
13             scanf("%d", &b[i][j]);
14         }
15     }
16     printf("\n\n");
17     //exibe
18     for(i=0;i<LINHA;i++){
19         for(j=0;j<COLUNA;j++){
20             //printf("%d\t", b[i][j]);
21             printf("[%d][%d]: %d\t", i,j,b[i][j]);|
22         }printf("\n");
23     }
24     printf("\n\n");
25     getchar();
26 }

```



Exercício (em sala)

- Leia uma matriz de 3x3 elementos inteiros e calcule a soma dos seus elementos



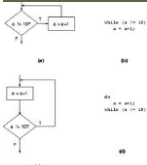
Exercício (em sala)

- Leia uma matriz de 3x3 elementos inteiros e calcule a soma dos seus elementos

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int mat[3][3];
    int i, j, soma = 0;
    printf("Digite os elementos da matriz\n");
    for(i=0; i < 3; i++)
        for(j=0; j < 3; j++) {
            scanf("%d", &mat[i][j]);
        }

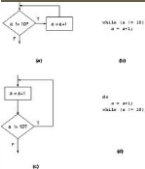
    for(i=0; i < 3; i++)
        for(j=0; j < 3; j++)
            soma = soma + mat[i][j];
    printf("Soma = %d\n", soma);

    return 0;
}
```



Exercício (em sala)

- Dado duas matrizes reais de dimensão 2x3, fazer um programa para calcular a soma delas.



Exercício (em sala)

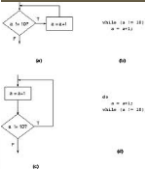
- Dado duas matrizes reais de dimensão 2x3, fazer um programa para calcular a soma delas.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    float A[2][3], B[2][3], S[2][3];
    int i, j;

    //leia as matrizes A e B...

    for(i=0; i < 2; i++)
        for(j=0; j < 3; j++)
            S[i][j] = A[i][j] + B[i][j];

    return 0;
}
```



Passagem de matrizes para funções

declaração da matriz na função principal:

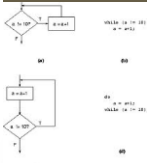
```
float mat[4][3]
```

protótipo da função (opção 1): parâmetro declarado como “vetor-linha”

```
void f (... , float (*mat)[3], ...);
```

protótipo da função (opção 2): parâmetro declarado como matriz, omitindo o número de linhas

```
void f (... , float mat[ ][3], ...);
```

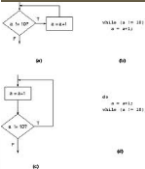


Exercício (fazer em sala)

- Criar funções para leitura e exibição de matrizes bidimensionais inteiras e reais de tamanho LINHA por COLUNA (constantes simbólicas), com os seguintes protótipos:

```
leMatriz_int(int matriz[][COLUNA], int linha, int coluna)
leMatriz_float(float matriz[][COLUNA], int linha, int coluna)

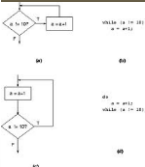
exibeMatriz_int(int matriz[][COLUNA],int linha, int coluna)
exibeMatriz_float(float matriz[][COLUNA],int linha, int coluna)
```
- Modificar as funções de exibição para exibir somente os valores (sem os índices)
- Fazer um menu solicitando ao usuário para mostrar a matriz com ou sem índice (usar uma função para cada exibição)



```

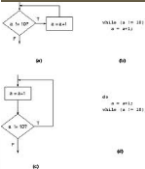
1  #include <stdio.h>
2  #include <stdlib.h>
3  /* Função : Exemplos de matrizes
4   Autor : Edkallenn - Data : 10/04/2012
5   Obs: */
6  #define LINHA 4
7  #define COLUNA 3
8  #define PULA printf("\n\n")
9
10 void lematriz(int b[][COLUNA], int, int);
11 void exibematriz(int b[][COLUNA], int, int);
12
13 main(){
14     int b[LINHA][COLUNA];          //declara
15     lematriz(b, LINHA, COLUNA);    //lê
16     PULA;
17     exibematriz(b, LINHA, COLUNA); //exibe
18     PULA;
19     getchar();
20 }
21 void lematriz(int mat[][COLUNA], int lin, int col){
22     int i, j;
23     for(i=0;i<lin;i++){            //linha
24         for(j=0;j<col;j++){        //coluna
25             printf("Digite o elemento mat[%d][%d]:", i+1, j+1);
26             scanf("%d", &mat[i][j]);
27         }
28     }
29 }
30 void exibematriz(int mat[][COLUNA], int lin, int col){
31     int i, j;
32     for(i=0;i<lin;i++){
33         for(j=0;j<col;j++){
34             printf("b[%d][%d]: %d\t", i, j, mat[i][j]);
35
36         }printf("\n");
37     }
38 }
39

```



Exercício (fazer em sala)

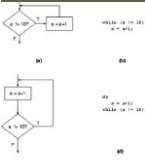
- Criar funções para preenchimento com valores aleatórios e posterior exibição de matrizes bidimensionais inteiras e reais de tamanho LINHA por COLUNA (constantes simbólicas).
- Usar as funções com valores iguais de LINHA e COLUNA (matrizes quadradas) a 10, 20, 50, 100, 1000



```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4 /* Função : Exemplos de matrizes - random
5 Autor : Edkallenn - Data : 10/04/2012
6 Obs: */
7 #define LINHA 10
8 #define COLUNA 10
9 #define PULA printf("\n\n")
10
11 void lematriz_random(int b[][COLUNA], int, int, int);
12 void exibematriz(int b[][COLUNA], int, int);
13 int random(int n);
14 main(){
15     int b[LINHA][COLUNA]; //declara
16     srand((unsigned)time(NULL)); //inicializa gerador de nos. aleatorios
17     puts("Gera matriz randomica");
18     PULA;
19     lematriz_random(b, LINHA, COLUNA, 100); //lê
20     exibematriz(b, LINHA, COLUNA); //exibe
21     getchar();
22 }
23 void lematriz_random(int mat[][COLUNA], int lin, int col, int n){
24     int i, j, valor;
25     for(i=0;i<lin;i++){ //linha
26         for(j=0;j<col;j++){ //coluna
27             valor = (1 + random(n-1));
28             mat[i][j]=valor;
29         }
30     }
31 }
32 void exibematriz(int mat[][COLUNA], int lin, int col){
33     int i, j;
34     for(i=0;i<lin;i++){
35         for(j=0;j<col;j++){
36             printf(" %d\t", mat[i][j]);
37
38             }printf("\n");
39     }
40 }
41 int random(int n){ //funcao para gerar aleatorios
42     return rand() % n;
43 }

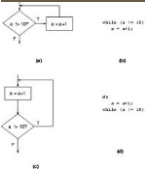
```



Manipulação com matrizes

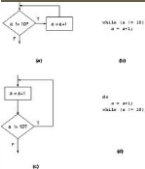
- Multiplicação por escalar:
- A matriz resultante consiste de todos os elementos multiplicados pelo escalar um a um
- EXERCÍCIO
- Fazer uma função para multiplicar uma matriz dada por um escalar informado pelo usuário (escalar inteiro). Protótipo:

```
mult_escalar(int matriz, int linha, int coluna, int escalar)
```



Manipulação com matrizes

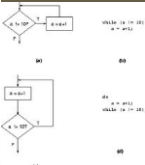
- O próximo exemplo usa uma **matriz bidimensional** para criar uma **grade** do jogo da velha.
- Inicialmente a matriz é preenchida com pontos
- Depois o programa forma um **ciclo** com um laço no qual a matriz é impressa, verifica se algum jogador já ganhou ou se houve empate, solicita que o jogador digite um par de coordenadas, atribui o caractere 'o' ou 'x' ao elemento da matriz correspondente às coordenadas entradas
- Os **índices** começam em zero.
- Vemos como trabalha uma matriz bidimensional



```

1  #include <stdio.h>
2  #include <stdlib.h>
3  /* Função : Jogo da velha - Viviane Mizrahi
4     Autor : Edkallenn - Data : 10/04/2012
5     Obs: */
6  #define PULA printf("\n")
7
8  main(){
9     unsigned char m[3][3];
10    int lin, col, j=0;
11
12    const int TRUE = 1;
13    const char O = 'o', X = 'x';
14
15    printf("Digite as coordenadas na forma lin, col: \n");
16    for(lin=0;lin<3;lin++)
17        for(col=0;col<3; col++)
18            m[lin][col]='.';
19
20    while(TRUE){
21        for(lin=0;lin<3;lin++)
22        {
23            for(col=0;col<3;col++)
24                printf("%c ", m[lin][col]);
25            PULA;
26        }
27        //verifica se o primeiro jogador ganhou
28        if((m[0][0]==O && m[0][1]==O && m[0][2]==O) ||
29           (m[1][0]==O && m[1][1]==O && m[1][2]==O) ||
30           (m[2][0]==O && m[2][1]==O && m[2][2]==O) ||
31           (m[0][0]==O && m[1][0]==O && m[2][0]==O) ||
32           (m[0][1]==O && m[1][1]==O && m[2][1]==O) ||
33           (m[0][2]==O && m[1][2]==O && m[2][2]==O) ||
34           (m[0][0]==O && m[1][1]==O && m[2][2]==O) ||
35           (m[0][2]==O && m[1][1]==O && m[2][0]==O))
36        {
37            printf("\aVocê ganhou, primeiro jogador!!!\n");
38            break;
39        }

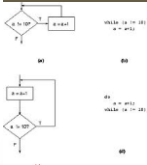
```



```

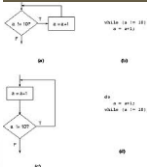
40 //verifica se o segundo jogador ganhou
41 if((m[0][0]==X && m[0][1]==X && m[0][2]==X) ||
42    (m[1][0]==X && m[1][1]==X && m[1][2]==X) ||
43    (m[2][0]==X && m[2][1]==X && m[2][2]==X) ||
44    (m[0][0]==X && m[1][0]==X && m[2][0]==X) ||
45    (m[0][1]==X && m[1][1]==X && m[2][1]==X) ||
46    (m[0][2]==X && m[1][2]==X && m[2][2]==X) ||
47    (m[0][0]==X && m[1][1]==X && m[2][2]==X) ||
48    (m[0][2]==X && m[1][1]==X && m[2][0]==X))
49 {
50     printf("\aVoce ganhou, segundo jogador!!!\n");
51     break;
52 }
53 if (j==9){
54     printf("\aEmpatou\n");
55     break;
56 }
57 printf("Coordenadas: ");
58 scanf("%d, %d", &lin, &col);
59
60 if(m[lin][col]== '.') //casa livre?
61 {
62     if(j%2) m[lin][col]=X;
63     else m[lin][col]=O;
64     j++;
65 }
66 }
67 PULA;
68 getchar();
69 }
70

```



Exercício

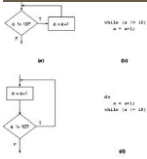
- Modifique o programa anterior para permitir somente entradas válidas.




```

1  #include <stdio.h>
2  #include <stdlib.h>
3  /* Função : Jogo da velha - Viviane Mizrahi
4     Autor : Edkallenn - Data : 10/04/2012
5     Obs: */
6  #define PULA printf("\n")
7
8  main(){
9     const char pt='.';
10    unsigned char m[3][3]=
11    {{pt, pt, pt}, {pt, pt, pt}, {pt, pt, pt}};
12    int lin, col, j=0;
13
14    const int TRUE = 1;
15    const char O = 'o', X = 'x';
16
17    printf("Digite as coordenadas na forma lin, col: \n");
18    while(TRUE){
19        for(lin=0;lin<3;lin++)
20        {
21            for(col=0;col<3;col++)
22                printf("%c ", m[lin][col]);
23            PULA;
24        }
25        //verifica se o primeiro jogador ganhou
26        if((m[0][0]==O && m[0][1]==O && m[0][2]==O) ||
27           (m[1][0]==O && m[1][1]==O && m[1][2]==O) ||
28           (m[2][0]==O && m[2][1]==O && m[2][2]==O) ||
29           (m[0][0]==O && m[1][0]==O && m[2][0]==O) ||
30           (m[0][1]==O && m[1][1]==O && m[2][1]==O) ||
31           (m[0][2]==O && m[1][2]==O && m[2][2]==O) ||
32           (m[0][0]==O && m[1][1]==O && m[2][2]==O) ||
33           (m[0][2]==O && m[1][1]==O && m[2][0]==O))
34        {
35            printf("\aVocê ganhou, primeiro jogador!!!\n");
36            break;
37        }

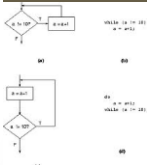
```



```

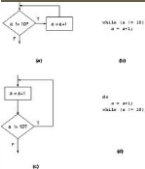
38 //verifica se o segundo jogador ganhou
39 if ((m[0][0]==X && m[0][1]==X && m[0][2]==X) ||
40      (m[1][0]==X && m[1][1]==X && m[1][2]==X) ||
41      (m[2][0]==X && m[2][1]==X && m[2][2]==X) ||
42      (m[0][0]==X && m[1][0]==X && m[2][0]==X) ||
43      (m[0][1]==X && m[1][1]==X && m[2][1]==X) ||
44      (m[0][2]==X && m[1][2]==X && m[2][2]==X) ||
45      (m[0][0]==X && m[1][1]==X && m[2][2]==X) ||
46      (m[0][2]==X && m[1][1]==X && m[2][0]==X))
47 {
48     printf("\aVoce ganhou, segundo jogador!!!\n");
49     break;
50 }
51 if (j==9){
52     printf("\aEmpatou\n");
53     break;
54 }
55 printf("Coordenadas: ");
56 scanf("%d, %d", &lin, &col);
57
58 if(m[lin][col]== '.') //casa livre?
59 {
60     if(j%2) m[lin][col]=X;
61     else m[lin][col]=O;
62     j++;
63 }
64 }
65 PULA;
66 getchar();
67 }
68

```



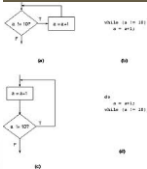
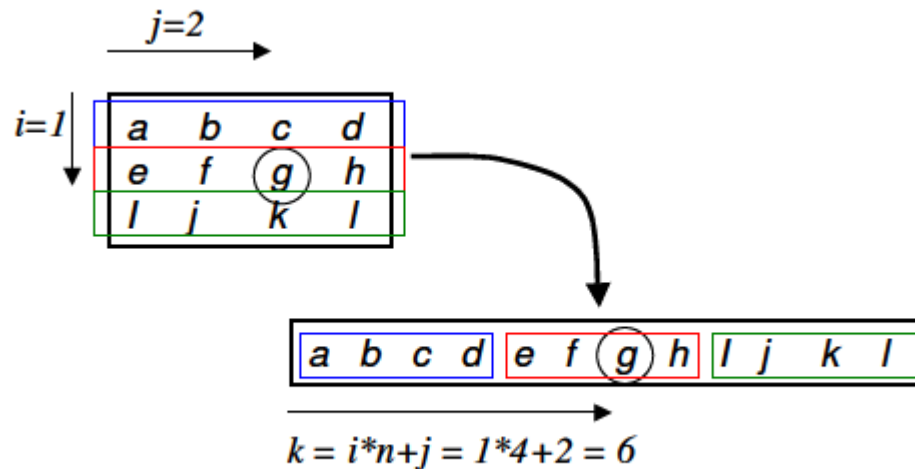
Matrizes dinâmicas

- Matriz representada por um **vetor simples**:
- conjunto **bidimensional** representado em vetor **unidimensional**
- estratégia:
 - primeiras posições do vetor armazenam elementos da primeira linha
 - seguidos dos elementos da segunda linha, e assim por diante
- **exige disciplina** para acessar os elementos da matriz



Matrizes dinâmicas

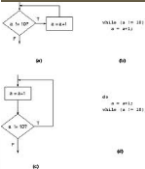
- Matriz representada por um vetor simples
- Matriz **mat** com n colunas representada no vetor **v**:
- **mat[i][j]** mapeado em **v[k]** onde **k = i * n + j**



Matrizes dinâmicas

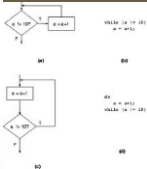
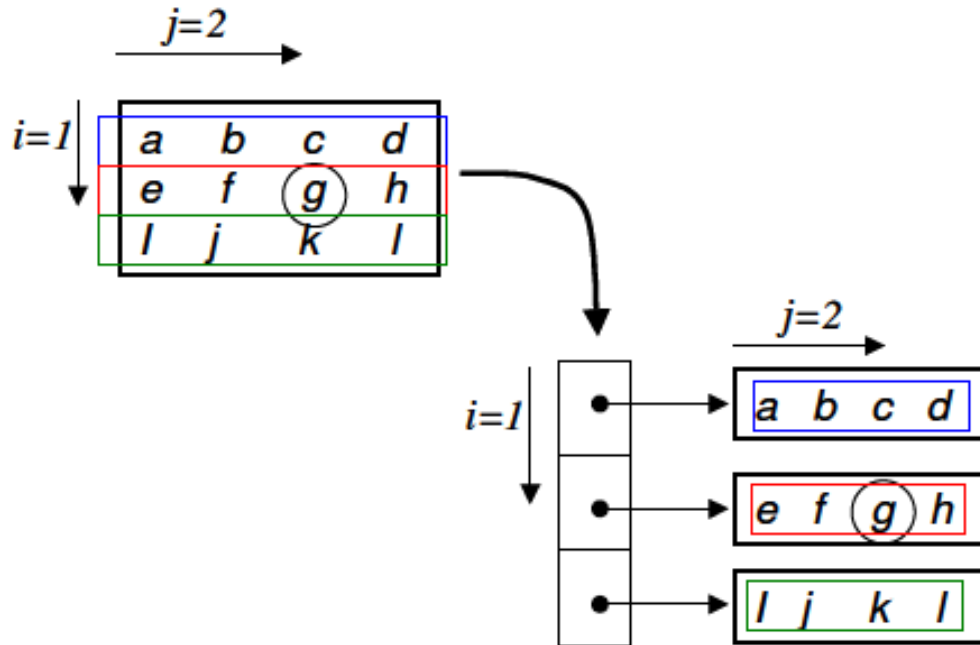
- Matriz representada por um vetor simples (cont.):
- `mat[i][j]` mapeado em `v[i * n + j]`

```
float *mat;    /* matriz m x n representada por um vetor */
...
mat = (float*) malloc(m*n*sizeof(float));
```



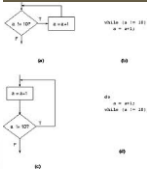
Matrizes Dinâmicas

- Matriz representada por um **vetor de ponteiros**:
- **cada elemento** do vetor **armazena o endereço** do **primeiro elemento** de cada linha da matriz



Operações com matrizes

- Exemplo – função transposta:
 - entrada: mat - matriz de dimensão m x n
 - saída: trp transposta de mat, alocada dinamicamente
 - Q é a *matriz transposta* de M se e somente se $Q_{ij} = M_{ji}$
- Solução 1: matriz alocada como vetor simples
- Solução 2: matriz alocada como vetor de ponteiros



```

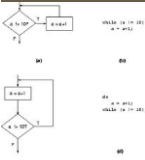
/* Solução 1: matriz alocada como vetor simples */
float* transposta (int m, int n, float* mat)
{
    int i, j;
    float* trp;

    /* aloca matriz transposta com n linhas e m colunas */
    trp = (float*) malloc(n*m*sizeof(float));

    /* preenche matriz */
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            trp[ j*m+i ] = mat[ i*n+j ];

    return trp;
}

```



```

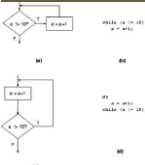
/* Solução 2: matriz alocada como vetor de ponteiros */
float** transposta (int m, int n, float** mat)
{
    int i, j;
    float** trp;

    /* aloca matriz transposta com n linhas e m colunas */
    trp = (float**) malloc(n*sizeof(float*));
    for (i=0; i<n; i++)
        trp[i] = (float*) malloc(m*sizeof(float));

    /* preenche matriz */
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            trp[ j ][ i ] = mat[ i ][ j ];

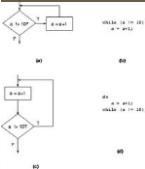
    return trp;
}

```



Alocação de arrays

- Para alocarmos arrays com mais de uma dimensão, utilizamos o conceito de “**ponteiro para ponteiro**”.
- Ex.: `char ***p3;`
- Para cada nível do ponteiro, fazemos a alocação de uma dimensão do array.



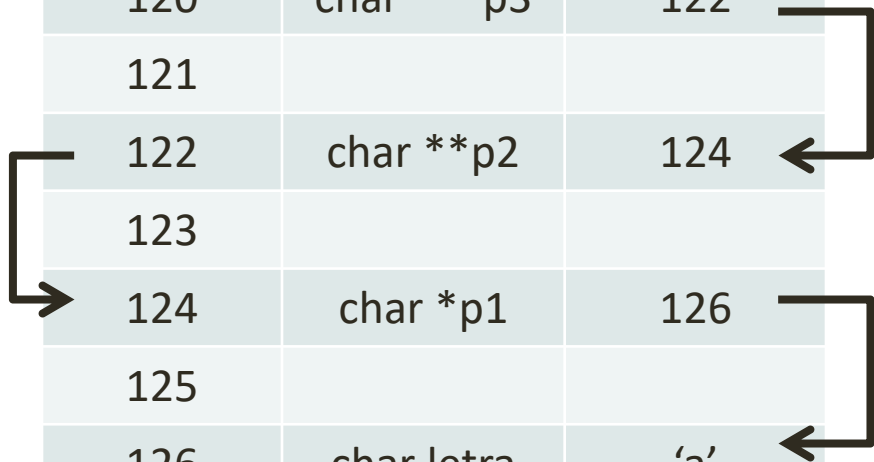
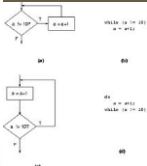
Alocação de arrays

- Conceito de “ponteiro para ponteiro”:

```
char letra = 'a';
char *p1;
char **p2;
char ***p3;
```

```
p1 = &letra;
p2 = &p1;
p3 = &p2;
```

Memória		
posição	variável	conteúdo
119		
120	char ***p3	122
121		
122	char **p2	124
123		
124	char *p1	126
125		
126	char letra	'a'
127		

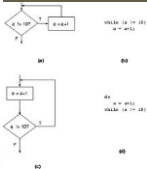
Alocação de arrays

- Em um ponteiro para ponteiro, cada nível do ponteiro permite criar uma nova dimensão no array.

```
int **p; //2 "*" = 2 níveis = 2 dimensões
int i, j, N = 2;
p = (int**) malloc(N*sizeof(int*));

for (i = 0; i < N; i++){
    p[i] = (int *)malloc(N*sizeof(int));
    for (j = 0; j < N; j++)
        scanf("%d", &p[i][j]);
}
```

Memória		
posição	variável	conteúdo
119	int **p	120
120	p[0]	123
121	p[1]	126
122		
123	p[0][0]	69
124	p[0][1]	74
125		
126	p[1][0]	14
127	p[1][1]	31
128		

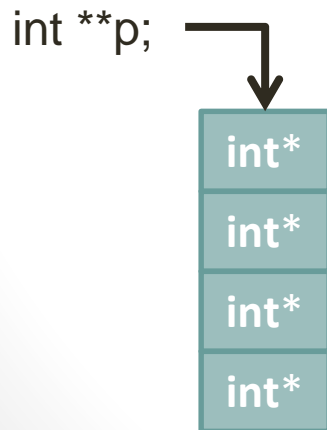


Alocação de arrays

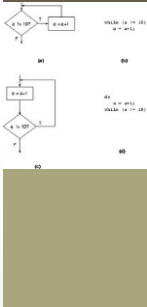
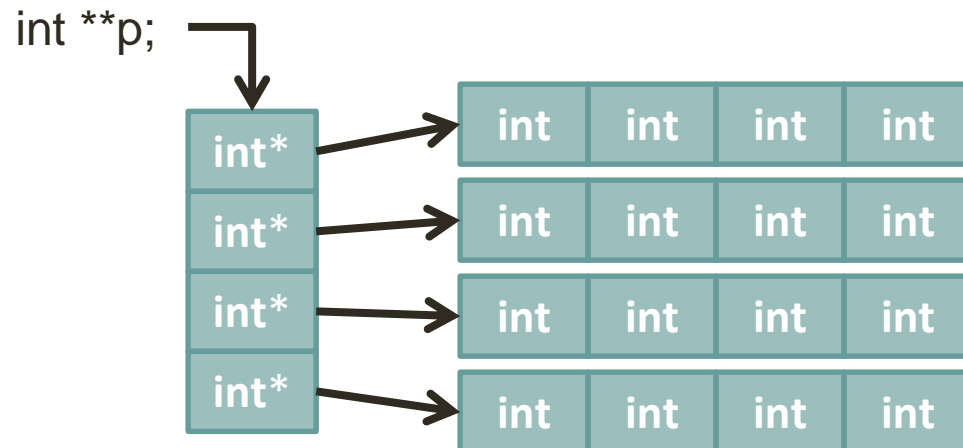
- Em um ponteiro para ponteiro, cada nível do ponteiro permite criar uma nova dimensão no array.

```
p = (int**) malloc(N*sizeof(int*));  
  
for (i = 0; i < N; i++){  
    p[i] = (int *)malloc(N*sizeof(int));  
}
```

1º malloc:
cria as linhas

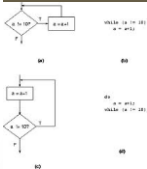


2º malloc:
cria as colunas



Alocação de arrays

- Diferente dos arrays de uma dimensão, para **liberar um array** com mais de uma dimensão da memória, é preciso **liberar a memória alocada em cada uma de suas dimensões**, na ordem inversa da que foi alocada

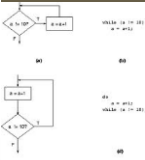


Alocação de arrays

```
int **p; //2 "*" = 2 níveis = 2 dimensões
int i, j, N = 2;
p = (int**) malloc(N*sizeof(int*));

for (i = 0; i < N; i++) {
    p[i] = (int *)malloc(N*sizeof(int));
    for (j = 0; j < N; j++)
        scanf("%d", &p[i][j]);
}
```

```
for (i = 0; i < N; i++)
    free(p[i]);
free(p);
```



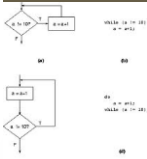
Alocação dinâmica arrays m-por-n

AlocacaoDinamicaArraysMulti.c VetoresMatrizes

```

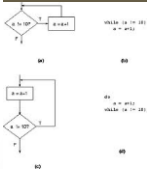
1  #include <stdio.h>
2  #include <stdlib.h>
3  /* Função : Alocação dinâmica de arrays multidimensionais
4     Autor : Edkallenn - Data : 06/04/2012 */
5  #define N 3 //tamanho maximo do array
6
7  main(){
8     int **p;
9     int i, j;
10    //cria e lê o array
11    p = (int**) malloc(N*sizeof(int)); //cria as linhas
12    for (i=0;i<N;i++){
13        p[i]=(int *) malloc(N*sizeof(int)); //cria as colunas
14        for (j=0;j<N;j++){
15            printf("Digite o elemento [%d][%d] do vetor: ",i,j);
16            scanf("%d", &p[i][j]);
17        }
18    }
19    printf("\n\nExibindo o array:\n\n");
20    //exibe o array
21    for(i=0;i<N;i++){
22        for(j=0;j<N;j++){
23            printf("[%d][%d] = %d\t", i,j,p[i][j]);
24        } printf("\n");
25    }
26
27    getch();
28 }
29

```



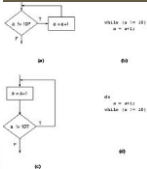
Matrizes de Ponteiros

- Uma matriz de strings é um array de ponteiros e é seu maior uso.



Matrizes de strings

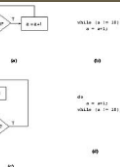
- Como uma string é uma matriz
- Uma matriz de strings é, na realidade, uma matriz de matrizes, ou matriz multidimensional (neste caso de duas dimensões)
- Examine o programa a seguir e verifique como funcionam as matrizes de strings na prática.

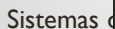


```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  /* Função : Imprime o dia da semana a partir de uma data
5  Autor : Edkallenn - Data : 10/04/2012
6  Obs: */
7  #define PULA printf("\n")
8  int dia_semana(int, int, int);
9  main(){
10     char diasemana[7][14]= {"Domingo",
11                             "Segunda-feira",
12                             "Terça-Feira",
13                             "Quarta-Feira",
14                             "Quinta-Feira",
15                             "Sexta-Feira",
16                             "Sabado"
17                             };
18
19     int dia, mes, ano;
20     const char ESC = 27;
21     do{
22         puts("Programa que exibe o dia da semana de uma data"); PULA;
23         printf("Digite a data na forma dd, mm, aaaa: ");
24         scanf("%d, %d, %d", &dia, &mes, &ano);
25         printf("\nO dia da semana desta data e: %s\n\n", diasemana[dia_semana(dia, mes, ano)]);
26         printf("ESC para terminar ou ENTER para recomencar\n\n");
27     }while (getch()!=ESC);
28     getch();
29     return 0;
30 }
31 int dia_semana( int dia, int mes, int ano){
32     int dSemana = ano + dia + 3 * (mes - 1) - 1;
33     if (mes<3)
34         ano--;
35     else
36         dSemana -= (int)(0.4*mes+2.3);
37     dSemana+= (int)(ano/4)- (int)((ano/100+1)*0.75);
38     dSemana %= 7;
39     return dSemana;
40 }

```





**VER A LISTA DE
EXERCÍCIOS QUE ESTARÁ
DISPONÍVEL NO BLOG, NO
DROPBOX OU NO URI.**

