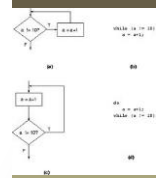
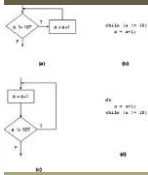
Estruturas de Dados

- 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/
  - Instagram: http://instagram.com/edkallenn ou @edkallenn
  - LinkedIn: br.linkedin.com/in/Edkallenn
  - Foursquare: https://pt.foursquare.com/edkallenn
- Telefones:
  - 68 98401-2103 (CLARO) 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



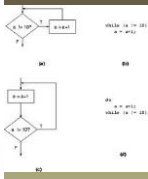
Agenda

- Vetores e Matrizes (arrays)
- Declaração
- Referenciando um elemento
- Armazenando dados no array (vetor ou matriz)
- Lendo dados
- Gerando um ponteiro para uma matriz
- Passando vetores para funções
- Matrizes bidimensionais
- Strings (Leitura, saída, inicializacao, funções de manipulação de strings - Introdução)
- Matrizes de strings
- Matrizes multidimensionais
- Indexando ponteiros



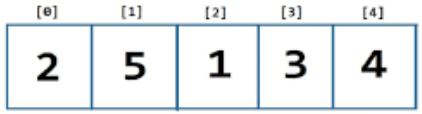
Matrizes e vetores(arrays)

- Uma **matriz (ou array)** é uma coleção de variáveis do mesmo tipo que é referenciada por um nome comum.
- São também conhecidas como arrays (ou arranjos)
- É uma *estrutura de dados homogênea* que consiste em itens de dados de um mesmo tipo, relacionados entre si.
- IMPORTANTE: Todos os elementos têm o mesmo nome e o mesmo tipo
- Acessamos um elemento específico mediante um **índice,** também conhecido como **subscrito**, ou **número da posição** do elemento na matriz.



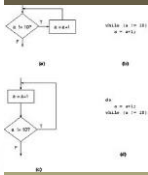
Matrizes e vetores (arrays)

- Em C todas os arrays consistem de posições contíguas de memória (posições adjacentes)
- O endereço mais baixo corresponde ao primeiro elemento e o mais alto ao último elemento
- Arrays podem ter de uma a várias dimensões.
- Um **vetor** é um caso específico de array unidimensional (uma única dimensão!)
- O array mais comum em C é a de **string** (que é um vetor de caracteres terminada por um nulo)



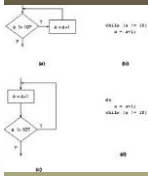
Matrizes e vetores (arrays)

- Em C, os arrays e os ponteiros estão intimamente relacionados
- Uma discussão sobre um deles normalmente refere-se ao outro
- Imagine um problema de calcular a média das notas da prova do mês de junho de 10 alunos.
- Você poderia declarar 10 variáveis diferentes
- Agora imagine se fossem 100, 300 ou 2000 alunos?
- Seria uma tarefa volumosa!
- É evidente que precisamos de uma maneira conveniente para referenciar tais coleções de dados similares.
- Precisamos de uma estrutura de dados, precisamos de uma matriz (na realidade, um vetor, que é um caso de matriz)



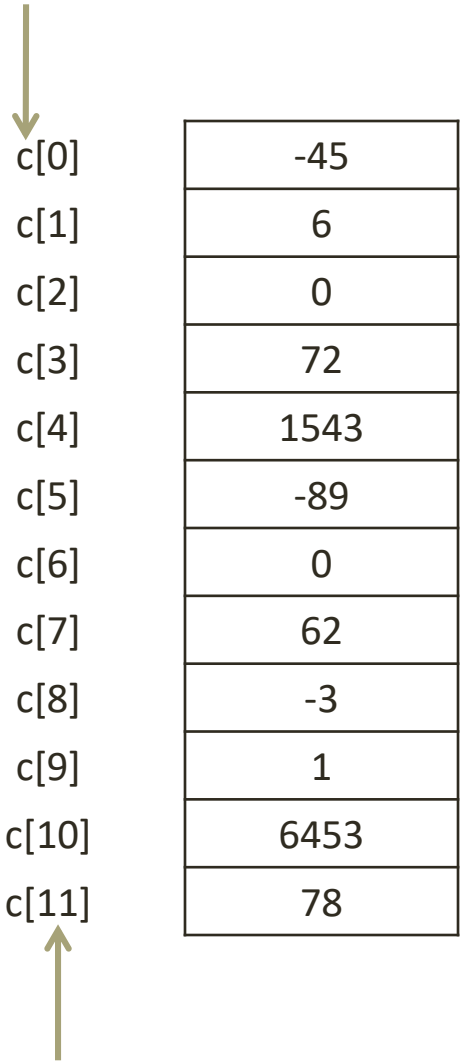
Matrizes e vetores (arrays)

- O próximo slide mostra uma matriz(array, vetor) de inteiros chamada **`c`**.
- Ela contém 12 elementos
- Pode-se fazer referência a qualquer elemento da matriz fornecendo o nome da matriz seguido do número da posição do elemento desejado entre colchetes(**`[ ]`**)
- O primeiro elemento em qualquer matriz (array) é sempre o **`elemento zero`**.
- O primeiro elemento de **`c`** é **`c[0]`**
- O segundo é **`c[1]`**, o sétimo é **`c[6]`**
- Em geral o elemento **`i`** do array (matriz) **`c`** é chamado **`c[i-1]`**
- O número entre colchetes (a posição do elemento) é chamada formalmente de subscrito ou índice.
- Um subscrito é um **`inteiro`** ou uma **`expressão inteira.`**



Uma matriz com doze elementos

Nome da matriz (observe que todos os elementos dessa matriz possuem o mesmo nome, c)



Número de posição do elemento dentro da matriz c

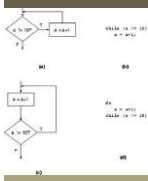
Matrizes e Vetores (arrays)

- Para dividir por 2 o sétimo elemento do array **c** e atribuir o resultado à variável **x**, escreveríamos:

```
x = c[6] / 2;
```

- Para imprimir a soma dos 3 primeiros elementos do array **c**, escreveríamos:

```
printf("%d", c[0] + c[1] + c[2]);
```

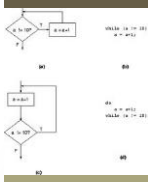


Declarando matrizes (arrays)

- A forma geral para declarar uma matriz unidimensional (vetor) é:

```
tipo nome_vetor[tamanho];
```

- Como outras variáveis as matrizes devem ser explicitamente **declaradas** para que o compilador possa alocar espaço na memória para elas.
- Aqui, **`tipo`** é o tipo base da matriz, que é o tipo de dados de cada elemento da matriz
- **`tamanho`** define até quantos elementos a matriz irá guardar



Exemplos de declaração

```
double balance[100];
```

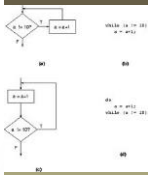
declara uma matriz (unidimensional, um vetor) de 100 elementos chamada balance, do tipo double.

```
int c[12];
```

declara uma matriz (unidimensional, um vetor) de 12 elementos chamada c, do tipo int.

```
char palavra[10];
```

declara uma matriz (de char – uma string) de 10 elementos chamada palavra, do tipo char.

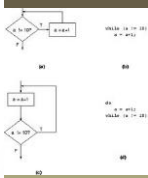


Observações Importantes

- Em C, toda matriz (array) tem 0 como índice do seu primeiro elemento.
- Portanto quando você escreve

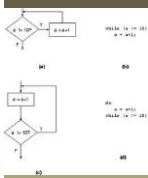
```
char palavra[10];
```

- Você está declarando um vetor de caracteres (uma string) que tem 10 elementos, `palavra[0]` até `palavra[9]`



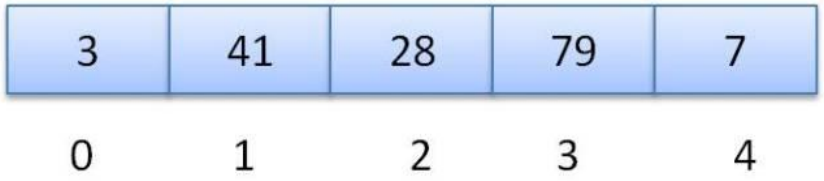
Exemplo de declaração

```c
#include <stdio.h>
#include <stdlib.h>
/*  Função : Segundo contato com arrays (vetores)
    Autor : Edkallenn
    Data : 06/04/2012
    Observações: arquivo-> declaracao_preenchimento.cpp
*/
#define MAX 100  //tamanho maximo do vetor

main(){
    int x[MAX];
    int t;

    // Preenche o vetor
    for (t=0;t<MAX;++t)
       x[t]=t; //forma normal - impares

    //Exibe
    for (t=0;t<MAX;t++)
        printf("%-3d ", x[t]);

    getchar();
}
```



Arrays - Características

- Estrutura homogênea, isto é, é formado por elementos do mesmo tipo.
- Todos os elementos da estrutura são igualmente acessíveis, isto é, o tempo e o tipo de procedimento para acessar qualquer um dos elementos do array são iguais.
- Cada elemento do array tem um índice próprio segundo sua posição no conjunto

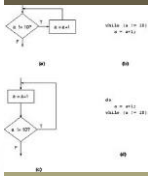


Quantidade de armazenamento

- A quantidade de armazenamento necessário para guardar um array está diretamente relacionada com seu tamanho e tipo
- Para uma matriz unidimensional(um vetor), o tamanho em bytes é calculado assim:

```
total em bytes = sizeof(tipo) * tamanho da matriz
```

- C não tem verificação implícita de limites em arrays. Você poderia ultrapassar os llimites e escrever nos dados de alguma outra variável.
- Cuidado com a verificação de limites.



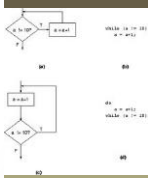
Inicialização de um vetor

- C permite inicialização de vetores (e matrizes) no momento da declaração.
- A forma geral de uma inicialização de um vetor é:

```
tipo nome_matriz[tamanho]={lista_valores}
```

- `lista_valores` é uma lista separada por vírgulas de constantes compatíveis com o `tipo`
- Ex:

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



Inicialização não dimensionada

- Exclui-se o tamanho e C declara com a quantidade inicializada.
- Ex:

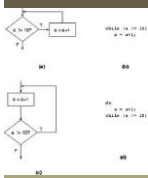
```
int a[]={1,2,5,6,-2,7};
```

- É semelhante a:

```
int a[6]={1,2,5,6,-2,7};
```

- Útil em strings (veremos a seguir):

```
char erro1[]="Erro de leitura\n";
```

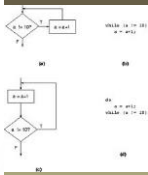


Inicialização de arrays

- Os elementos de um array(matriz) podem ser inicializados de três maneiras: por declaração, por atribuição e por entrada de valores (input)
- Se houver menos inicializadores do que o tamanho do array, os elementos restantes são incializados com o valor zero.
- Ex:

```
int n[10] = {0};
```

- Inicializa explicitamente o primeiro elemento com zero e automaticamente inicializa os restantes com 0.
- O programador deve inicializar pelo menos um elemento com zero para que os elementos restantes sejam automaticamente zerados.



Inicializar com dados (input)

declaracao_preenchimento_com_input.cpp VetoresMatrizes

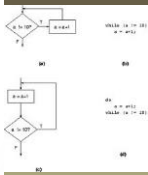
```
1   #include <stdio.h>
2   #include <stdlib.h>
3   /*  Função : Inicializacao com inputs (vetores)
4       Autor : Edkallenn
5       Data : 06/04/2012
6       Observações: arquivo-> declaracao_preenchimento_com_input.cpp
7   */
8   #define MAX 10  //tamanho maximo do vetor
9   main(){
10      int x[MAX];
11      int i, t;
12
13      // Preenche o vetor
14      for (i=0;i<MAX;++i){
15          printf("\nDigite o elemento %d do vetor: ", i);
16          scanf("%d", &x[i]);
17      }
18      //Exibe
19      printf("\nO vetor digitado eh\n");
20      for (t=0;t<MAX;t++)
21          printf("%-3d ", x[t]);
22
23      getchar();
24  }
```

Soma dos elementos de um vetor

```c
#include <stdio.h>
#include <stdlib.h>
/*  Função : Soma de elementos de um vetor sem funcoes
    Autor : Edkallenn
    Data : 06/04/2012
    Observações:
*/
#define MAX 12

main(){
    //incializacao do vetor vet. Poderia ter sido vet[] = { valores };
    int i, vet[MAX] = {-45, 6, 0, 72, 1543, -89, 0, 62, -3, 1, 6453, 78};
    int j, s=0, soma;

    for(i=0;i<MAX;i++){  //Exibe o vetor
        printf("O elemento %d do vetor eh: %d\n", i, vet[i]);
    }
    for(j=0;j<MAX;j++){ // Soma os elementos
        s+=vet[j];
    }
    soma=s;

    printf("\n\nA soma dos elementos do vetor eh: %d\n\n", soma);
    printf("A media dos elementos do vetor eh: %d\n\n", soma/MAX);

}
```

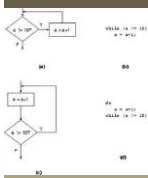
Erros comuns

- Esquecer de inicializar os elementos de um array que precisam ser inicializados
- Fornecer mais inicializadores para uma matriz do que o número de seus elementos é um erro de sintaxe
- Fazer referência a um elemento além dos limites de um array (matriz)



Práticas recomendadas

- Use apenas letras maiúsculas para nomes de constantes simbólicas (declarações #DEFINE)
- Em loops de matrizes (arrays), os subscritos (índices) nunca devem ser menores que zero e sempre devem ser menores do que o número total de elementos do array (tamanho – 1).
- Certifique-se que a condição de término do loop evita o acesso a elementos fora desses limites
- Definir o tamanho de cada matriz (array) como uma constante simbólica torna os programas mais flexíveis



Inicializar e exibir com procedimentos

```c
#include <stdio.h>
#include <stdlib.h>
/*  Função : Inicializacao com inputs (vetores)
    Autor : Edkallenn  - Data : 06/04/2012
    Observações:
*/
#define MAX 10  //tamanho maximo do vetor

void preenche_vetor(void);
void exibe_vetor(void);
int x[MAX];

main(){
    preenche_vetor();
    exibe_vetor();
}

void preenche_vetor(void){  // Preenche o vetor
    int i;
    for (i=0;i<MAX;++i){
        printf("\nDigite o elemento %d do vetor: ", i);
        scanf("%d", &x[i]);
    }
}
void exibe_vetor(void){     //Exibe
    int t;
    printf("\nO vetor digitado eh\n");
    for (t=0;t<MAX;t++)
        printf("%-3d ", x[t]);
}
```

Qual o problema deste programa???

Gerando um ponteiro para um array

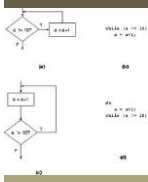
- Pode se gerar um ponteiro para o primeiro elemento de um array simplesmente especificando o nome da matriz, sem nenhum índice. Assim, dado o vetor:

```
int exemplo[10];
```

- Pode se gerar um ponteiro para o 1º elemento usando somente o nome **exemplo**. Este fragmento atribui a **ptr** o endereço do primeiro elemento de **exemplo**:

```
int *ptr;
int exemplo[10];
ptr = exemplo;
```

- Pode-se, é claro, usar o operador **&** para os mesmos efeitos. Desta forma, **exemplo** e **&exemplo[0]** produzem os mesmos resultados. (prefere-se sempre a primeira forma)

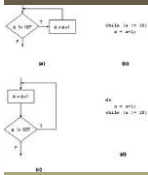


Lembrando a aula passada

```c
#include <stdio.h>
#include <stdlib.h>
/*  Função : Programa para imprimir histograma
    Autor : Edkallenn
    Data : 06/04/2012
    Observações: Programa interessante para imprimir histograma!
*/
#define TAMANHO 10  //tamanho maximo do vetor

main(){
    int n[TAMANHO] = {19, 3, 15, 7, 11, 9, 13, 5, 17, 1};
    int i, j;

    printf("%s%13s%17s\n", "Elemento", "Valor", "Histograma");

    for(i = 0;i<TAMANHO;i++)
    {
        printf("%8d%13d        ", i, n[i]);   // 07 espacos em branco

        for(j = 1;j<=n[i];j++)    //Imprime uma barra de asteriscos
            printf("%c", '*');

            printf("\n");

    }
    //no DevC++ coloque aqui system("pause"); ou getchar();
    return 0;
}
```

Passando arrays para funções

- Unidimensionais (vetores)
- Em C você **não pode** passar uma matriz inteira (array) como argumento para uma função.
- Você **pode passar um ponteiro** para um array para uma função especificando o nome do array sem um índice.
- Ou seja, para passar, como um argumento, uma array(matriz) para uma função, **especifique o nome do array sem colocar colchetes**

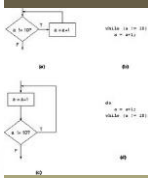


Passagem de arrays à funções

- Exemplo:

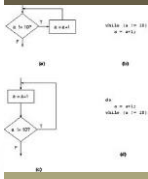
```
void main(void){
        int exemplo[10];
        func1(exemplo);
        .
        .
        .
}
```

- O trecho acima passa o endereço de **exemplo** para **func1()**
- **O endereço de exemplo[0] ou &exemplo[0]**



Passagem de arrays à funções

- A linguagem C passa automaticamente os arryas às funções usando chamadas por referência simuladas
- As funções chamadas podem, portanto, modificar os valores dos elementos nos arrays originais dos locais que fazem as chamadas
- Na verdade, o nome de um array é o endereço do primeiro elemento do array!
- Quando a função chamada modifica os elementos do arrayem seu corpo, os elementos REAIS estão sendo modificados em suas posições originais de memória

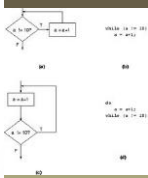


Parâmetros formais/reais de arrays (vetores)

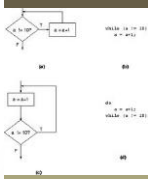
- Se uma função recebe um vetor, pode se declarar o parâmetro formal de 3 formas, como um ponteiro, com uma matriz dimensionada ou como uma matriz não dimensionada
- Exemplo: **função func1()**

```
void func1(int *x)      //ponteiro
void func1(int x[10])   //array dimensionado
void func1(int x[])     //array não-dimensionado
```

- Os 3 métodos produzem resultados idênticos, pois todos dizem ao compilador que um ponteiro será recebido.
- Como C não verifica limites, o tamanho não importa.



```c
#include <stdio.h>
#include <stdlib.h>
/*  Função : Inicializacao com inputs (vetores)
    Autor : Edkallenn  - Data : 06/04/2012
    Observações: Usa a passagem por referencia do primeiro elemento
    do vetor como parametro. Demonstra como o relacionamento entre ponteiros e
    matrizes é estreito em C.
*/
#define MAX 5  //tamanho maximo do vetor

void preenche_vetor(int []);    //prototipo das funcoes (procedimentos)
void exibe_vetor(int []);

main(){
    int x[MAX]; //vetor
    preenche_vetor(x);
    exibe_vetor(x);
}

void preenche_vetor(int vet[]){  // Preenche o vetor
    int i;
    for (i=0;i<MAX;++i){    //quaisquer alteracoes aqui afetam x[MAX] (referencia)
        printf("\nDigite o elemento %d do vetor: ", i);
        scanf("%d", &vet[i]);
    }
}
void exibe_vetor(int v[]){     //Exibe
    int t;
    printf("\nO vetor digitado eh\n");
    for (t=0;t<MAX;t++)
        printf("%-3d ", v[t]);
}
```



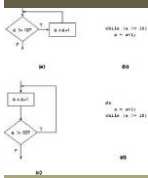
Importante

- Embora as matrizes sejam passadas por referência, os elementos individuais são passados por valor exatamente como variáveis simples.
- Para passar UM elemento da matriz para uma função, use o nome do elemento da matriz com o subscrito (índice) como argumento.
- O protótipo de uma função que recebe uma matriz é:

```
tipo nome_funcao(tipo nome_qualquer_matriz[])
```

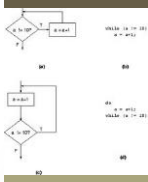
ou

```
tipo nome_funcao(tipo [])
```



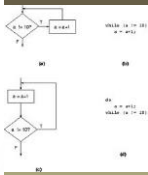
Remove elementos de um vetor

- A operação de remoção de elementos de um vetor `v[0..n-1]` consiste em retirar o elemento que tem o índice `k` e fazer com que o vetor resultante tenha índices `0,1,...,n-2`.
- Por exemplo, o resultado da remoção do elemento de índice `3` no vetor `000,111,222,333,444,555` é o vetor `000,111,222,444,555`).
- A operação, é claro, só faz sentido `se 0 <= k < n.`
- A próxima função faz isso...



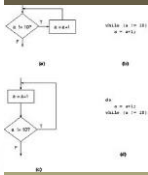
Função remove(int k, int v[], int n)

```c
int remove_elemento(int k, int v[], int n){
/*
remove o elemento de índice k do vetor v[0..n-1] e devolve
o novo valor de n. A função supõe, claro, que 0<=k<n
*/
    int j;
    for(j=k;j<n-1;j++){
        v[j]=v[j+1];
    }
    return n-1;
}
```



Exemplo

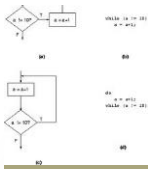
- Programa para calcular a média e a variância (ambas como funcões) de uma classe de 10 alunos.
- Próximo slide



```c
#include <stdio.h>
#include <stdlib.h>
/*  Função : Média e variância de 10 numeros reais (vetores)
    Autor : Edkallenn  - Data : 06/04/2012 */
#define MAX 10  //tamanho maximo do vetor
void preenche_vetor(float []);    //prototipo das funcoes (procedimentos)
void exibe_vetor(float []);
float media(int n, float *v);     //prototipo alternativo(mostra uso ponteiros)
float variancia(int n, float *v, float med);
main(){
    float med, var, x[MAX];       //media, variancia e vetor original
    preenche_vetor(x);
    med = media(MAX, x);
    var = variancia(MAX, x, med);
    exibe_vetor(x);
    printf("\n\nA Media = %g e a Variancia = %g \n", med, var);
    return 0; //no devC++ antes desta coloque system("pause"); ou getchar();
}
float media(int n, float *v){   //funcao que calcula a media de um vetor
    float soma=0.0f;            //de n elementos
    for (int i=0;i<n;i++)
        soma+=v[i];
    return soma/n;
}
float variancia(int n, float *v, float med){ //calcula a variancia dada a media
    float s = 0.0f;                          //que eh passada como parametro
    for (int i = 0;i<n;i++)                  //alem do tamanho n do vetor
        s+=((v[i]-med)*(v[i]-med));
    return s/n;
}
```

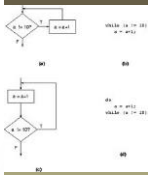
Continuação (a partir da linha 31)

```
31  void exibe_vetor(float v[]){      //Exibe o vetor do tipo float
32      int t;
33      printf("\nO vetor digitado eh\n\n");
34      for (t=0;t<MAX;t++)
35          printf("%-3g ", v[t]);
36  }
37  void preenche_vetor(float vet[]){   // Preenche o vetor
38      int i;
39      for (i=0;i<MAX;++i){ //quaisquer alteracoes aqui afetam x[MAX] (referencia)
40          printf("\nDigite o elemento %d do vetor: ", i);
41          scanf("%g", &vet[i]);
42          }
43  }
44
```



Mais um exemplo (o elemento máximo de um vetor)

- Considere o seguinte problema:
  - Determinar o valor de um elemento máximo de um vetor v[0 .. n-1] .
- É claro que o problema só faz sentido se o vetor não é vazio, ou seja, se n ≥ 1.
- Para preparar o terreno, examine uma tradicional solução iterativa do problema:



Máximo Iterativa

```c
int maximo( int tamanho, int v[])
{
    int j, maximo;
    maximo = v[0];
    for (j = 1; j < tamanho; j += 1){
        if (v[j] > maximo)
            maximo = v[j];
    }
    return maximo;
}
```

- Arquivo

Programa completo de teste

```c
#include<stdio.h>
#define MAX 10

int maximo (int, int*);

int maximo( int tamanho, int v[])
{
   int j, maximo;
   maximo = v[0];
   for (j = 1; j < tamanho; j += 1){
      if (v[j] > maximo)
        maximo = v[j];
   }
   return maximo;
}
main(){
    int maior, vetor[MAX]={1,20,3,4,5,6,20,8,9,10};

    maior = maximo(10, vetor);
    printf("O elemento maximo do vetor eh: %d\n", maior);
    getch();
}
```

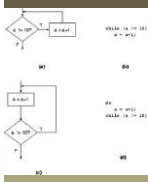
Solução Recursiva do problema

- Eis uma função recursiva que resolve o problema da seção anterior:

```
int maximo_r( int tamanho, int v[]){
    if (tamanho == 1)
        return v[0];
    else{
        int x;
        x = maximo_r(tamanho-1, v);  /* máximo de v[0..n-2] */
        if (x > v[tamanho-1])
            return x;
        else
            return v[tamanho-1];
    }
}
```

Análise

- A análise do algoritmo tem a mesma forma que uma prova por indução.
- Se n vale 1 então v[0] é o único elemento relevante do nosso vetor e portanto v[0] é o máximo.
- Agora suponha que n vale mais que 1.
- Então nosso vetor tem duas partes: v[0..n-2] e v[n-1] e portanto o valor que procuramos é o maior dentre
    - v[n-1] e o máximo de v[0..n-2].



Insere Elemento

```
int insere_elemento(int indice, int elemento, int v[], int n){
/*  insere o elemento elemento entre as posições indice-1 e indice do vetor
    e devolve o novo valor de n. Supõe que 0 <= indice <= n
*/
    int j;
    for(j=n;j>indice;j--){
        v[j]=v[j-1];
    }
    v[indice] = elemento;
    return n + 1;
}
```

insercao_vetor.c

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

#define MAX 10  //tamanho maximo do vetor
void preenche_vetor(int, int z[MAX]);    //prototipo das funcoes (procedimentos)
void exibe_vetor(int, int []);
int remove_elemento(int k, int v[], int n);
int insere_elemento(int k, int y, int v[], int n);

int main(){
    int indice, novo_tamanho, elemento, x[MAX];
    preenche_vetor(MAX, x);
    exibe_vetor(MAX, x); printf("\n");
    printf("Remove elemento\n");
    printf("Qual o indice do elemento que vc quer remover? ");
    scanf(" %d", &indice);
    novo_tamanho = remove_elemento(indice, x, MAX);
    printf("O novo tamanho e´: %d\n", novo_tamanho);
    exibe_vetor(novo_tamanho, x); printf("\n");
    printf("Insere elemento\n");
    printf("Qual elemento e em qual posicao que vc quer inserir? ");
    scanf(" %d %d", &elemento, &indice);
    insere_elemento(indice, elemento, x, novo_tamanho);
    exibe_vetor(MAX, x);
    getchar(); //no devC++ antes desta coloque system("pause"); ou getchar();
}
```

Preenchendo vetor com valores aleatórios

- A linguagem C dispõe da função rand() para a geração de números pseudo-aleatórios
- Ela gera um inteiro aleatório entre 0 e RAND_MAX (constante definida em stdlib.h)
- Podemos usar a seguinte função random(int n):

```
int random(int n){
    return rand() % n;
}
```

- Esta função retorna o resto da divisão entre o valor gerado por rand() e n, ou seja, retorna um inteiro pertencente ao intervalo [0, n-1]

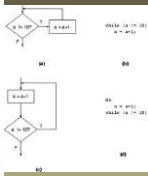
Preenchendo vetor com valores aleatórios

- Exemplos:

```
int numero_produtos = 1 + random(200);
```

- Gera uma quantidade aleatória entre 1 e 200 e armazena na variável numero_produtos
- Os números gerados são pseudo-aleatórios (explicar).
- É interessante que a cada execução do programa novos números sejam gerados.
- Para isso deve se fornecer uma "semente" para a função rand()
- Para uma dada "semente", a função rand() gera uma determinada sequência de números
- A semente, deve, portanto ser diferente.
- Uma boa ideia é usar a função time() (cabeçalho time.h) que retorna o número de segundos transcorridos desde 01/janeiro/1970 e exige um parametro para armazenar o valor de retorno. Se não for preciso guardar o valor de retorno, usa-se NULL como parâmetro.
- A função srand exige um unsigned como parâmetro. Podemos usar este valor para reiniciar a "semente" da função rand. Assim:

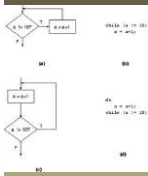
```
srand((unsigned)time(NULL));
```



Teste de random()

re | le_exibe_com_procedimento.cpp | media_variancia_10_reais.cpp | AlocacaoDinamicaA

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include<time.h>
4   /*  Função : Teste de random
5   **  Autor  : Edkallenn
6   **  Data : 06/04/2012
7   */
8   int random(int n);
9
10  main(){
11      srand((unsigned)time(NULL));
12      int i;
13      printf("Alguns numeros randomicos\n");
14      for(i=1;i<=10;i++){
15          printf("%d\t", 1 + random(10));
16      }
17      getchar();
18  }
19  int random(int n){
20      return rand()%n;
21  }
22
```



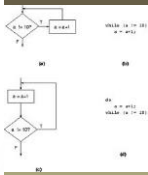
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100  //tamanho maximo do vetor
/*  Função : Preenche vetor com numeros aleatorios (vetores)
    Autor : Edkallenn  - Data : 06/04/2012
    Observções: Usa a função rand() e srand para gerar numeros aleatarios
*/
void preenche_vetor_random(int []);   //prototipação das funcoes
void exibe_vetor_int(int []);
int aleatorio(int n);

void exibe_vetor_int(int* v){      //Exibe o vetor
    int t;
    printf("\nO vetor gerado pelo computador eh:\n\n");
    for (t=0;t<MAX;t++)
        printf("%-3d\t", v[t]);
}
void preenche_vetor_random(int vet[]){  // Preenche com valores randomicos
    int i, valor;
    for (i=0;i<MAX;++i){
        valor = (aleatorio(200));   //gera ate 200
        vet[i]=valor;
    }
}
int aleatorio(int n){      //funcao para gerar aleatorios
    return rand() % n + 1;
}

int main(){
    int x[MAX];            //vetor original
    srand(time(NULL));    //inicializa gerador de nos. aleatorios
    preenche_vetor_random(x);
    exibe_vetor_int(x);
    getchar();
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 100  //tamanho maximo do vetor
/*  Função :Gera numeros aleatorios do tipo float
    Autor : Edkallenn  - Data : 06/04/2012
    Observções: Usa a função rand() e srand para gerar numeros aleatários
*/
void preenche_vetor_aleatorio(float []);   //prototipo das funcoes
void exibe_vetor_float(float *);
int aleatorio(int n);

main(){
    float x[MAX];                       //vetor original
    srand(time(NULL));    //inicializa gerador de nos. aleatorios
    preenche_vetor_aleatorio(x);
    exibe_vetor_float(x);
    getchar();
    return 0;
}
void exibe_vetor_float(float v[]){      //Exibe o vetor
    int t;
    printf("\nO vetor digitado eh\n");
    for (t=0;t<MAX;t++)
        printf("\t%4.2f ", v[t]);
}
void preenche_vetor_aleatorio(float vet[]){  // Preenche com valores aleatorios
    int i;                                   // Do tipo FLOAT
    float valor, num;
    for (i=0;i<MAX;++i){
        num = 1 + aleatorio(200);  //gera divisor para gerar float
        valor = (1 + (10/num)) + aleatorio(200-1); //gera ate 100
        vet[i]=valor;
    }
}
int aleatorio(int n){      //funcao para gerar aleatorios
    return rand() % n;
}
```


Descobrir a maior e a menor com #define

- Problema: ler as notas obtidas pelos alunos de uma classe e determinar: a maior nota obtida, a menor nota obtida, a média das notas e número de alunos com nota abaixo da média.
- Vamos usar uma macro para realizar a condição de obter a maior e a menor nota, assim:

```
#define MAX(x,y) (((x) > (y)) ? (x) : (y))
#define MIN(x,y) (((x) < (y)) ? (x) : (y))
```

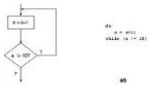


```c
#include <stdio.h>
#include <stdlib.h>
#define TAM 10  //tamanho maximo do vetor
#define MAX(x,y) (((x) > (y)) ? (x) : (y))
#define MIN(x,y) (((x) < (y)) ? (x) : (y))
/*  Função :Ler notas, dizer a maior, a menor, a media e quantas abaixo
**    Autor : Edkallenn  - Data : 06/04/2012
**   Observações: Usa macro define para achar o maior e o menor
*/
void preenche_vetor_float(int n, float []);   //prototipo das funcoes
void exibe_vetor_float(int n, float vet[TAM]);
float media(int n, float *v);

main(){
    int j, abaixo=0;        //vetor original
    float notas[TAM], maior, menor, med;  //e outras variaveis
    maior =-1; menor = 11;
    preenche_vetor_float(TAM, notas);
    exibe_vetor_float(TAM, notas);
    med = media(TAM, notas);
    for(j=0;j<TAM;j++){
        maior = MAX(maior, notas[j]);
        menor = MIN(menor, notas[j]);
        if(notas[j]<med)
            abaixo++;
    }
```

```
27      printf("\nA media eh: %5.2f\nA maior nota eh: %5.2f", med, maior);
28      printf("\nA menor eh: %5.2f\nAbaixo da media: %d\n\n", menor, abaixo);
29      getchar();
30      return 0;
31  }
32  void exibe_vetor_float(int n, float v[]){       //Exibe o vetor de floats
33      int t;
34      printf("\nO vetor digitado eh\n\n");
35      for (t=0;t<n;t++)
36          printf("\t%5.2f\t", v[t]);
37  }
38  void preenche_vetor_float(int n, float vet[]){  // Preenche o vetor com float
39      int i;
40      for (i=0;i<n;i++){
41          printf("\nDigite o elemento %d do vetor: ", i);
42          scanf("%f", &vet[i]);
43      }
44  }
45  float media(int n, float *v){    //funcao que calcula a media de um vetor
46      float soma;
47      soma =0.0f;
48      int i;
49      for (i=0;i<n;i++) soma+=v[i];
50      return soma/n;
51  }
```

Exercício (fazer em sala)

- Reescrever o programa da média e da variância de 10 números reais preenchendo agora o vetor com valores randômicos.
- Perguntar ao usuário se ele quer usar preencher o vetor com inteiros ou com reais (float) e preencher com o tipo solicitado
- Executar o programa com MAX valendo 100, 200, 300, 10 e 20
- Usar todas as funções anteriores


- VER A LISTA DE EXERCÍCIOS QUE ESTARÁ DISPONÍVEL NO BLOG E NO DROPBOX.
- VERIFIQUE TAMBÉM SE NÃO HÁ EXERCÍCIOS NO URI ou no Google Class