Vetores e Strings

Disciplina de Programação de Computadores I
Universidade Federal de Ouro Preto

Agenda

- Vetores
- Strings
- Exercícios

Motivação para o uso de Vetores

- Frequentemente, utilizamos poucas variáveis nos programas.

- Quando desejamos utilizar diversas variáveis de um mesmo tipo, pode ser inviável declarar cada uma das variáveis.

- Quando queremos armazenar uma quantidade de variáveis definida em função de requisição do usuário, não temos como declarar estas variáveis no momento da codificação.

Definição de Vetores

- Vetor é uma coleção de variáveis do mesmo tipo referenciadas por um nome comum.
- Vetores:
    - Permitem acesso por meio de um índice inteiro;
    - São criados (alocados) em posições contíguas na memória;
    - Possuem tamanho (dimensão) pré-definido, ou seja, o tamanho de um vetor não pode ser alterado durante a execução do programa;
    - Índices fora dos limites causam comportamento imprevisível no programa.

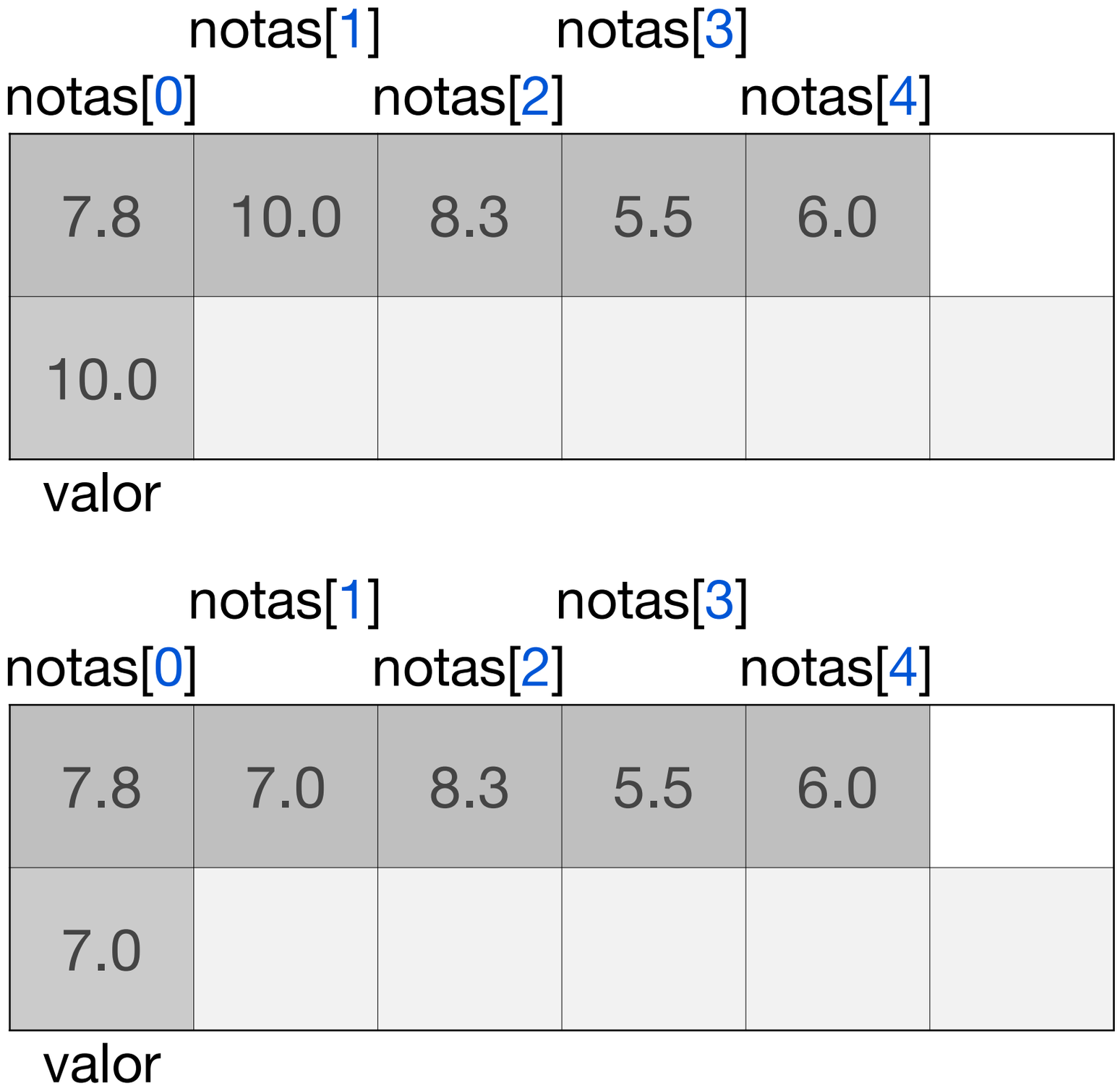
Declaração de Vetor

tipo_vetor nome_do_vetor [tamanho_do_vetor];

- tamanho_do_vetor é um número inteiro ou uma variável do tipo int (ver http://gcc.gnu.org/onlinedocs/gcc/Variable-Length.html).
- Esta declaração cria tamanho_do_vetor variáveis do tipo tipo_vetor.
- As variáveis criadas pelo vetor são acessadas por:
  - nome_do_vetor[0]
  - nome_do_vetor[1]
  - ...
  - nome_do_vetor[tamanho_do_vetor -1]

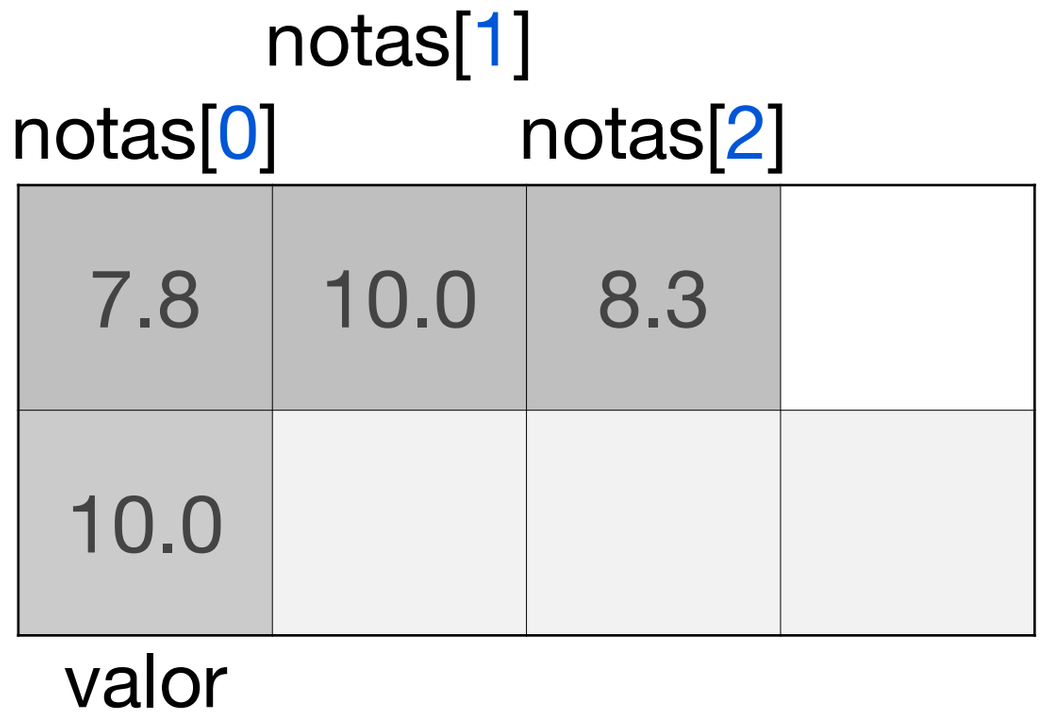
Exemplo de Vetor com índices inteiros

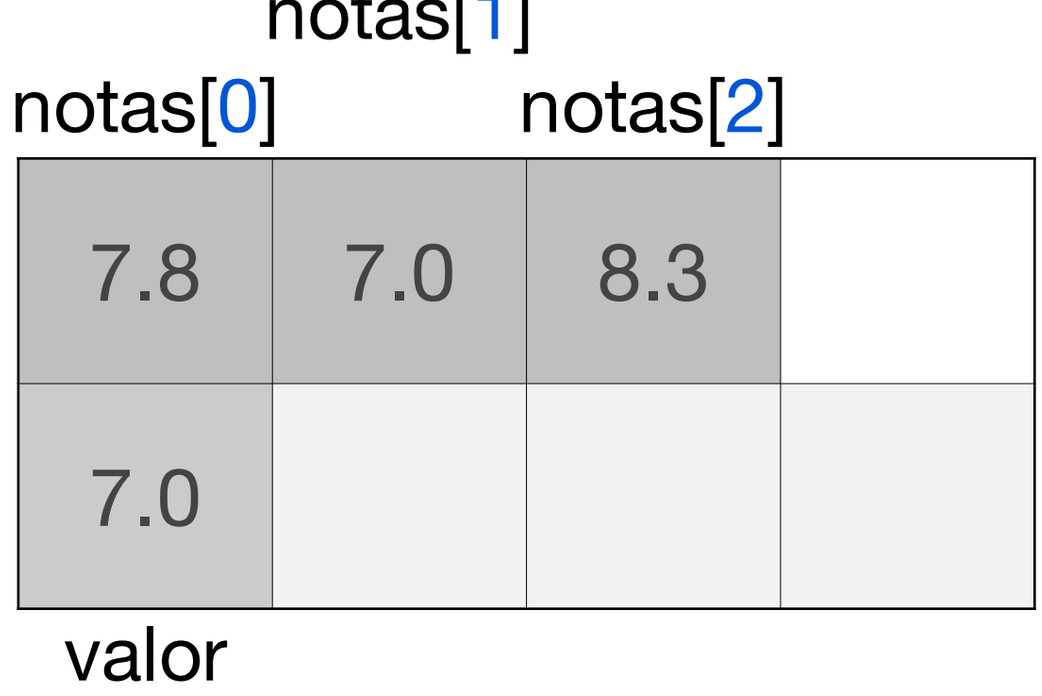
```
float notas [ 5 ];

notas[0] = 7.8;

notas[1] = 10.0;

notas[2] = 8.3;

notas[3] = 5.5;

notas[4] = 6.0;

float valor = notas[1];

valor = valor - 3;

notas[1] = valor;
```



Exemplo de Vetor com variáveis nos índices

```
int tamanho = 3;

int indice = 0;

float notas [ tamanho ];

notas[indice++] = 7.8;

notas[indice++] = 10.0;

notas[indice++] = 8.3;

float valor = notas[1];

valor = valor - 3;

notas[1] = valor;
```

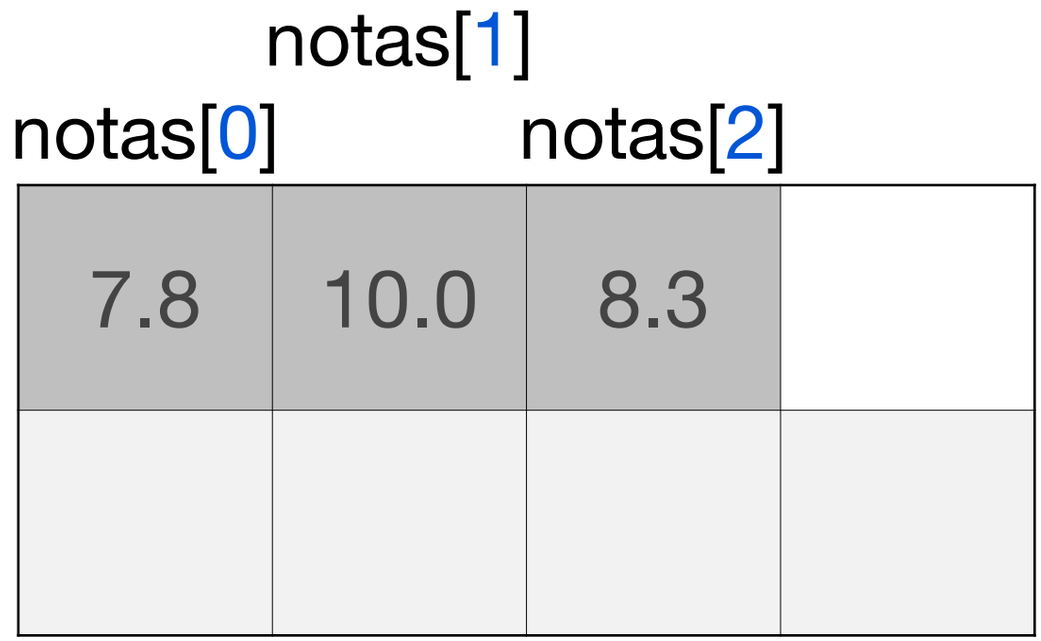




Acesso a índice fora do limite

```
int tamanho = 3;

int indice = 0;

float notas [ tamanho ];

notas[0] = 7.8;

notas[1] = 10.0;

notas[2] = 8.3;

notas[3] = 9.0;
```

// Este acesso altera uma posição de memória indevida e pode causar Segmentation Fault se a posição de memória não estiver reservada para o programa atual.



Ler, preencher e imprimir um vetor

```
int i, tamanho =5;
float notas [ tamanho ];
for (i=0; i< tamanho; i++){
        printf(“Digite o valor da posição notas[%d]:\n”,i);
        scanf(“%f”, &notas[i]);
}
printf(“Valores lidos:\n”);
for (i=0; i< tamanho; i++){
        printf(“notas[%d] = %.2f\n”, i, notas[i]);
}
```

Exemplo: Produto interno de vetores

- Crie 2 vetores de dimensão 5, leia os valores para estes vetores e calcule o produto interno destes vetores.

- Produto interno de dois vetores é a soma dos produtos entre os elementos em posições equivalentes dos vetores.

- Por exemplo, o produto interno dos vetores:
  <2,3,4,5,6> e <3,5,6,1,5>
  é igual a
  2*3 + 3*5 + 4*6 + 5*1+ 6*5

Código: Produto interno de vetores

```
int main(void){
    int i; double vetor1[5], vetor2[5], resultado;
    for(i=0; i<5; i++){
        printf("Digite o valor para vetor1[%d]:\n",i);
        scanf("%lf", &vetor1[i]);
    }
    for(i=0; i<5; i++){
        printf("Entre com valor para vetor2[%d]:\n",i);
        scanf("%lf", &vetor2[i]);
    }
    resultado = 0.0;
    for(i=0; i < 5; i++)
        resultado = resultado + ( vetor1[i] * vetor2[i] );
    printf("\nO produto interno é: %lf\n", resultado);
    return 0;
}
```

Vetores como parâmetros de sub-rotinas

- Vetores podem ser passados como parâmetros em sub-rotinas.

- Ao se passar um vetor como parâmetro, deve-se passar, também, o seu tamanho.

- Ao se passar um vetor como parâmetro de sub-rotina, **não é criado um novo vetor** local na sub-rotina.

- Isto significa que **os valores de um vetor são alterados dentro de uma sub-rotina!**

Vetores como parâmetros de sub-rotinas: exemplo

```
void proc5(int vet[], int tam){
        int i;
        for(i=0; i<tam; i++)
           vet[i]=5;
}
int main(void){
        int i, tamanho = 10, x[tamanho];
        for(i=0; i< tamanho; i++)
           x[i]=8;
        proc5(x, tamanho);
        for(i=0; i< tamanho; i++)
           printf("%d\n", x[i]);
        return 0;
}
```

Vetores e retornos de sub-rotinas

- Vetores não podem ser retornados por sub-rotinas

- Pode-se utilizar o fato de que vetores são alterados dentro de sub-rotinas para simular o retorno de um vetor por uma sub-rotina:
    - basta que um procedimento receba o vetor e o altere o seu conteúdo.

- O conteúdo será visto por quem chamou com as alterações.

Vetores como parâmetros de sub-rotinas: exemplo

```
void leVet(int vet[], int tam){
        int i;
        printf("Digite %d numeros: ",
                tam);
        for(i = 0; i < tam; i++)
                scanf("%d", &vet[i]);
}
void escreveVet(int vet[], int tam){
        int i;
        for(i=0; i< tam; i++)
           printf("vet[%d] = %d\n", i,
vet[i]);
}
int main(int){
        int vet1[10], vet2[20];
        leVet(vet1,10);
        leVet(vet2,20);
        escreveVet(vet1,10);
        escreveVet(vet2,20);
        return 0;
}
```

Inicialização de vetores

- Assim como variáveis dos demais tipos, vetores podem ser declarados e inicializados ao mesmo tempo.

tipo nome [ ] = { elementos separados por vírgula }

Exemplos:

```
double vet1[ ] = {2.3, 3.4, 4.5, 5.6};
int vet2[ ] = {5, 4, 3, 10, -1, 0};
```

- O tamanho do vetor é definido pela quantidade de elementos na lista entre parênteses.

Exemplo de declaração e inicialização de vetores

```
int main(void){
        double vet1[] = {2.3, 3.4, 4.5, 5.6};
        int vet2[] = {5, 4, 3, 10, -1, 0};
        int i;
        for(i=0; i<4; i++)
                printf("%lf\n", vet1[i]);
        for(i=0; i<6; i++)
                printf("%d\n", vet2[i]);
        return 0;
}
```

Cadeias de Caracteres

- A linguagem C não possui o tipo string (cadeia de caracteres) explicitamente, mas podemos considerar um vetor de caracteres como uma string.
- Em C, uma string é sempre terminada pelo caractere especial: '\0'
- Logo, ao declararmos um vetor de caracteres, devemos somar 1 à quantidade desejada de caracteres!
- Exemplo:
  char st1[7] = "string";

Declaração e inicialização de cadeiras de caracteres

- Podemos declarar e inicializar um vetor de caracteres de duas formas:

```
char st1[ ] = “string”;
char st2[ ] = {‘s’, ‘t’, ‘r’, ‘i’, ‘n’ , 'g’, '\0'};
```

- O tamanho da variável será a quantidade de caracteres atribuídos MAIS UM, devido ao caractere ‘\0’.

Leitura e impressão de cadeias de caracteres

- Uma cadeia de caracteres é lida ou impressa com o modificador %s

- O armazenamento da leitura é interrompido ao se encontrar um espaço, mesmo que existam mais caracteres depois do espaço.

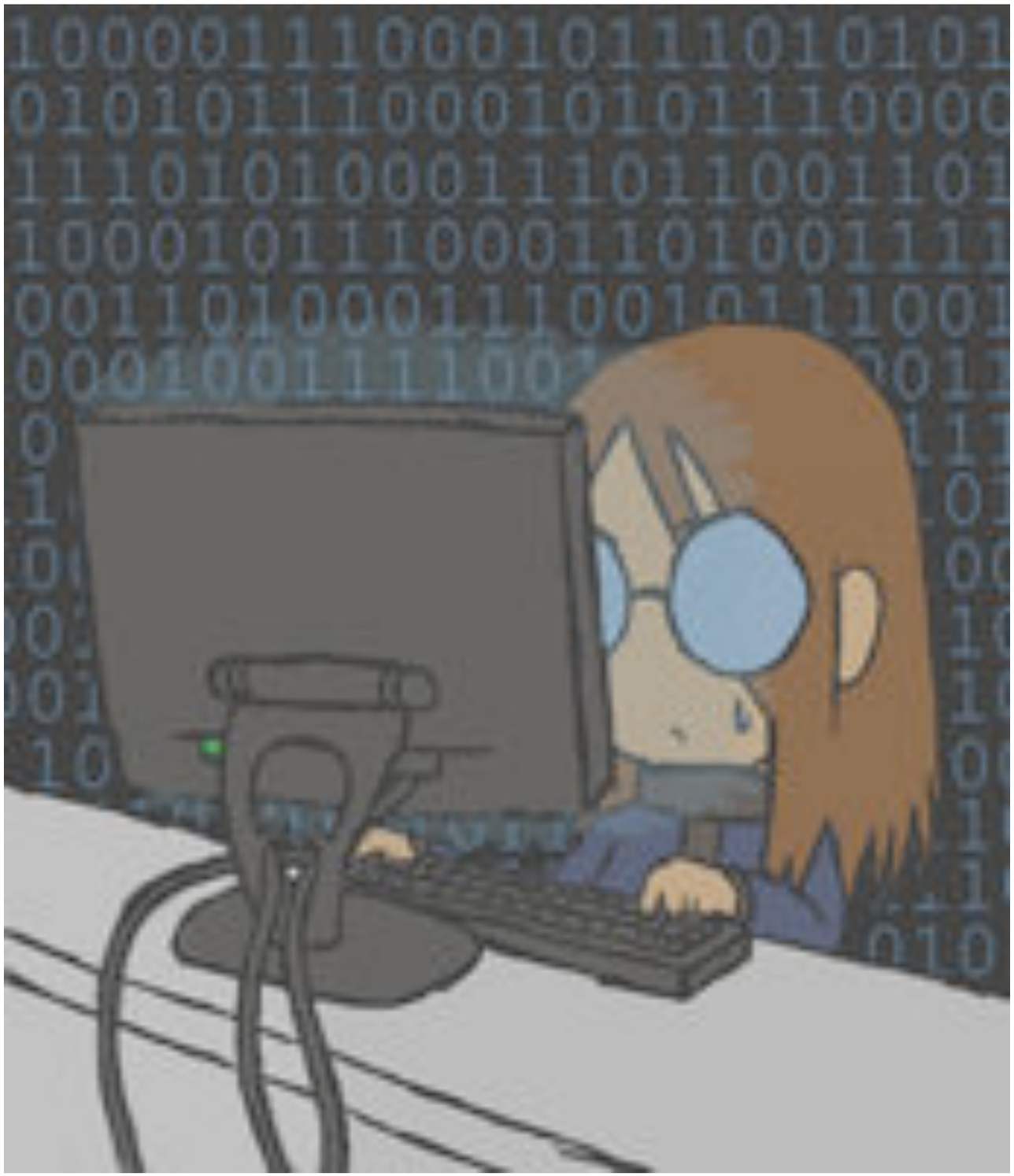
- Para ler uma cadeia de caracteres contendo espaços, indique que a cadeia deve terminar com a quebra de linha, assim: %[^\n]s

- A impressão da cadeia de caracteres é feita até o último caractere antes de ‘\0’

Cópia de strings

- Strings podem ser copiadas através da função strcpy da biblioteca string.h

```
char st1[ ] = “string”;

char st2[10];

char st3[31];

strcpy(st2, st1);

strcpy(st3, “Programacao de Computadores 1”);
```



Exercícios

Exercício

- Crie uma função com a assinatura:

```
int maiorValor(int vetor[ ], int tamanho);
```

- que receba como parâmetros um vetor e seu tamanho e devolva o maior valor armazenado no vetor.

Exercício

- Crie uma função com a assinatura:

```
int ordenado(int vetor[ ], int tamanho);
```

- que receba como parâmetros um vetor e seu tamanho e retorne:
  - 1, se os valores no vetor estiverem em ordem decrescente;
  - -1, se os valores no vetor estiverem em ordem crescente;
  - 0, caso contrário.

Exercício

- Crie uma função com a assinatura:

```
double media(int vetor[ ], int tamanho);
```

- que receba como parâmetros um vetor e seu tamanho e retorne a média dos valores armazenados no vetor.

Exercício

- Crie uma função com a assinatura:

  int palindrome(int vetor[ ], int tamanho);

- que receba como parâmetros um vetor e seu tamanho e retorne:
  - 1, se os valores no vetor formam um palíndrome;
  - 0, caso contrário.
- Um vetor é palíndrome se a sequência de valores do menor para o maior índice é igual à sequência de valores do maior para o menor índice
- Por exemplo, <1,2,3,4,3,2,1> é palíndrome, mas <1,2,3,1,2,3> não é.

Exercício

- Crie uma função com a assinatura:

```
int verifica(int vetor[ ], int tamanho, int produto);
```

- que receba como parâmetros um vetor e seu tamanho e um outro inteiro e retorne:
  - 1, caso caso existam dois elementos distintos do vetor tal que a multiplicação destes resulte em produto;
  - 0, caso contrário.
- Exemplo:
  Se vetor = <2, 4, 5, –10, 7> e C = 35, então o retorno é 1.
  Se vetor = <2, 4, 5, –10, 7> e C = –1, então o retorno é 0.

Exercício

- Escreva um programa que leia uma string de até 40 caracteres, armazene-a em um vetor e imprima a sua inversa na tela.

Solução com while

```
int main(){
        char st[40], stInv[40];
        int i, j, tam = 0;
        printf("Digite a string: ");
        scanf("%s", st);

        while( st[tam] != '\0'
                && tam < 40 ){
            tam++;
        }
        stInv[tam] = '\0';
        j = tam-1;
        i = 0;
        while( i < tam ){
                stInv[j] = st[i];
                i++;
                j--;
        }
        printf("Inversa: %s\n",
stInv);
 }
```

Solução com for

```
int main(){
        int i, j, tam = 0;
        char st[40], stInv[40];
        printf("Digite a string:\n");
        scanf("%s", st);
        for(tam=0 ; (st1[tam] != ‘\0’) && (tam < 40) ; tam++);
        stInv[tam] = ‘\0’;
        for( j = tam - 1, i = 0 ; j >= 0 ; j--, i++ ){
                stInv[ j ] = st1[ i ];
        }
        printf("Inversa: %s\n", stInv);
}
```

Solução sem vetor auxiliar

```
int main(){
        int i, j, tam = 0; char st1[40], aux;
        printf("Digite um texto (max. 40):");
        scanf("%s",st1);
        while(st1[tam] != ’\0’ && tam < 40)
          tam++;
        i = 0; j = tam -1;
        while(i < j){
          aux = st1[i];
          st1[i] = st1[j];
          st1[j] = aux;
          i++; j--;
        }
        printf("Inversa: %s\n",st1);
}
```

Referências Bibliográficas

- Material de aula do Prof. Ricardo Anido, da UNICAMP: http://www.ic.unicamp.br/~ranido/mc102/

- Material de aula da Profa. Virgínia F. Mota: https://sites.google.com/site/virginiaferm/home/disciplinas

- DEITEL, P; DEITEL, H. *C How to Program*. 6a Ed. Pearson, 2010.