

# Algoritmos e programação

**Prof. Dr. Josenalde Barbosa de Oliveira**

josenalde@eaj.ufrn.br

Aulas: 24T345 – 6 CRDS – 90h

# Variáveis indexadas (vetores, arrays unidimensionais)

- ▶ Conjuntos homogêneos, ou seja, com o mesmo tipo de dado
- ▶ Ao invés da declaração de uma única variável, tem-se a declaração de um conjunto de variáveis do mesmo tipo, que pode ter cada valor individual manipulado por meio de um **índice** (idx)

```
int x, y;  
x = 20; // atribuição  
y = x;  
// acesso ao valor de x
```

```
int x[5], y;  
x[2] = 20;  
/* atribuição do valor 20 ao índice 2  
do vetor x */  
y = x[2]; // acesso ao índice 2 de x
```

- ▶ Em C++, o primeiro índice é 0, logo o vetor x acima tem 5 elementos, com índices de 0 a 4. Ou seja, se o tamanho do vetor é N, os índices vão de 0 a N-1.

índices:		0	1	2	3	4
x:				20		

- Um vetor pode ser declarado e inicializado com uma lista de valores

```
int x[5] = {1,4,7,10,15};    int x[] = {1,4,7,10,15};  
                                long estudantes[500] = {0}; // todos 0  
                                long estudantes[500] = {}; // todos 0
```

- Se a lista é inicializada, não é obrigatório definir o tamanho e pode-se usar `[]`, pois o compilador infere a partir da lista.

## índices:

<b>x:</b>	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>Valores:</b>	1	4	7	10	15

- Já a instrução `float y[];` por exemplo não é válida, pois não há inicialização.

```
int N,i;  
cin>>i>>N;  
int v[N];  
if (i<N) {  
    v[i]=i;  
    cout << v[i] << endl;  
} else cout << "fora dos limites do array";
```

- Lendo um conjunto de valores. Primeira linha a quantidade de itens a ler e, na segunda os N itens são inseridos.

```
int N;  
cin>>N;  
int v[N], i{0};  
while (i < N) {  
    cin>>v[i++]; //incrementa i e depois usa como índice do vetor  
    cout<<v[i-1];  
}
```

- Exemplo: ler N números, salvar num vetor, e informar o maior número:

```
maior = v[0];  
for (int i=1; i<N; i++)  
    if (v[i]>maior)  
        maior = v[i];  
cout<<maior<<endl;
```


# Variáveis indexadas (vetores, arrays unidimensionais)

► Percorrer um vetor (ordem crescente e inversa), calcular media e outras estatísticas são tarefas comuns e básicas com vetores.

```
float v[5];
float mA, S=0;
for (int i=0; i<5; i++) {
    cin>>v[i];
    cout<<v[i];
}
mA = S / 5;
cout<<mA;
return 0;
```

```
bool flag = false;
// para somar apenas diferentes
float v[5];
float mA, S=0;
for (int i=0; i<5; i++) {
    cin>>v[i];
    if (i>0) {
        for (int j=i-1; j>=0; j--) {
            if (v[i] == v[j]) {
                flag=true;
                i-=1;
                break;
            }
        }
        cout<<v[i];

        if (!flag) S += v[i]; flag = 0;
    }
} ...
```



# Variáveis indexadas (vetores, arrays unidimensionais)

## ► Código básico para ordenação

```
int v[5] = {3,0,-1,1,2};  
for (int i=0;i < 4;i++) {  
    if (v[i]>v[i+1]) {  
        int aux=v[i];  
        v[i] = v[i+1];  
        v[i+1] = aux;  
        i = -1; // reset  
    }  
}  
for (int i=0;i < 5;i++)  
    cout<<v[i]<<"\t";  
return 0;  
}
```

Para crescente >

Para decrescente <

Algoritmo da TROCA (swap)

**Vetor ordenado:**

**-1**

**0**

**1**

**2**

**3**

# Ordenação otimizada

► O Código básico não é eficiente, pois a cada troca, retorna ao início do vetor. Para tamanhos maiores, por exemplo, maiores que 1000, pode ser inviável. C e C++ possuem uma função chamada `qsort` que implementa o método Quicksort, bem mais eficiente, pois trabalha com particionamento de vetor (dividir para conquistar)

Exemplo:

```
int fcomp(int *n1, int *n2) {  
    if (*n1 < *n2) return 1;  
    else return -1;  
    return 0;  
}  
  
int x[5] = {1,10,-1,2,10};  
  
qsort(x,5,sizeof(int),fcomp); }
```

► Exemplo: ler N números, salvar num vetor, e informar o maior número:

`qsort` – nome da função (`cstdlib`)

`x` – nome do vetor a ordenar

O terceiro parâmetro é o tamanho de cada elemento, em bytes

`fcomp` é o nome da função externa de comparação

# Ordenação otimizada – classe `vector`

➡ A STL (Standard Template Library) do C++ inclui uma série de classes para manipular tipos estruturados de dados, como vetores, listas, filas, pilhas etc.

➡ A função usa o conceito de iterador (iterator) para marcar o início (`begin()`) e fim (`end()`)

➡ A função `sort` está em `<algorithm>`

```
bool fcomp(int n1, int n2) {  
    return (n1 > n2); //crescente  
}
```

```
#include <iostream>  
#include <algorithm>  
#include <vector>  
using namespace std;  
  
int main() {  
  
    vector<int> v1; //tamanho zero, sem elementos  
    int n;  
    cin >> n;  
    vector<double> v2(n); //tamanho n dinâmico  
    cout<<"Valores do vetor: \n";  
    for (int i=0;i<n;i++) {  
        cin>>v2[i];  
    }  
    for (int i=0;i<n;i++) {  
        cout<<v2[i]<<'\\t';  
    }  
    sort(v2.begin(), v2.end()); //crescente  
    for (int i=0;i<n;i++) {  
        cout<<v2[i]<<'\\t';  
    }  
    cout << "\\nFIM" << endl;
```



# Ordenação otimizada (s18.cpp)

```
#include <ctime>
#include <cstdlib> // para o rand()
#define MIN      (1.50f)
#define MAX      (4.50f)

int comp(double *, double *);
void ordenaDec(double [], unsigned int);

int main() {
    srand(time(NULL));
    clock_t start_t, end_t;
    unsigned int sV = 100000;
    double x[sV];
    for (int i=0;i<sV;i++) {
        x[i] = MIN + (rand() / (RAND_MAX / (MAX - MIN)));
    }
}
```

# Ordenação otimizada

```
start_t = clock();
//qsort (x, sV, sizeof(double), comp);
ordenaDec(x, sV); // ordenação simples, com reset
end_t = clock();
double total_t = (double)(end_t - start_t) / (double)CLOCKS_PER_SEC;
cout.precision(14);
cout<<std::fixed << "Execution time CPU: " << total_t;
return 0;
}
}

int fcomp(double *n1, double *n2) { // by Josenalde
    if (*n1 < *n2) return 1;
    else return -1;
    return 0;
}
```

# Ordenação otimizada

```
void ordenaDec(double v[], unsigned int tam) {  
    double aux;  
    for (int i=0;i<tam-1;i++) {  
        if (v[i] < v[i+1]) {  
            aux = v[i];  
            v[i] = v[i+1];  
            v[i+1] = aux;  
            i = -1;  
        }  
    }  
}
```

**Esta é a função mais simples e didática para ordenação, porém ineficiente.**

**Reset do contador, volta ao início do vetor**

**Se  $a < b$ , Decrescente**  
**Se  $a > b$ , Crescente**

# Problema 01 com vetores

**Peça perdida** (OBI 2007, P1, F1):

<https://olimpiada.ic.unicamp.br/pratique/p1/2007/f1/perdida/>

Joãozinho adora quebra-cabeças, essa é sua brincadeira favorita. O grande problema, porém, é que às vezes o jogo vem com uma peça faltando. Isso irrita bastante o pobre menino, que tem de descobrir qual peça está faltando e solicitar uma peça de reposição ao fabricante do jogo. Sabendo que o quebra-cabeças tem  $N$  peças, numeradas de 1 a  $N$  e que exatamente uma está faltando, ajude Joãozinho a saber qual peça ele tem de pedir.

**Tarefa:** Escreva um programa que, dado um inteiro  $N$  e  $N - 1$  inteiros numerados de 1 a  $N$ , descubra qual inteiro está faltando.

**Entrada:** A entrada contém um único conjunto de testes, que deve ser lido do *dispositivo de entrada padrão* (normalmente o teclado). A entrada contém 2 linhas. A primeira linha contém um inteiro  $N$  ( $2 \leq N \leq 1.000$ ). A segunda linha contém  $N - 1$  inteiros numerados de 1 a  $N$  (sem repetições).

**Saída:** Seu programa deve imprimir, na *saída padrão*, uma única linha, contendo o número que está faltando na sequência dada.

# Problema 01 com vetores

**Peça perdida (OBI 2007, P1, F1):**

**Entrada:**

3

3 1

5

1 2 3 5

4

2 4 3

**Saída:**

2

4

1

## Problema 02: análise de pseudocódigo

Analise o algoritmo abaixo, verificando a saída para a seguinte sequência de pares de  $x$  e  $y$ , que devem estar num vetor com 10 elementos:

**10 e 5**

**26 e 13**

**30 e 22**

**2 e 1**

**36 e 10**

**Início**

**Enquanto** ( $y \neq 0$ ) **Faça**

$r \leftarrow x \bmod y$

$x \leftarrow y$

$y \leftarrow r$

**Fim\_Enquanto**

**Exiba**( $x$ )

O professor de história precisa dividir uma turma de alunos em grupos, de modo que cada grupo tenha a mesma quantidade de alunos. Nessa turma temos 24 alunas e 16 alunos. Quantos componentes terá cada grupo? (MDC, ALGORITMO DE EUCLIDES)

O piso de uma sala retangular, medindo  $3,52 \text{ m} \times 4,16 \text{ m}$ , será revestido com ladrilhos quadrados, de mesma dimensão, inteiros, de forma que não fique espaço vazio entre ladrilhos vizinhos. Os ladrilhos serão escolhidos de modo que tenham a maior dimensão possível. Na situação apresentada, o lado do ladrilho deverá medir:

## Problema 03:

Projete o algoritmo e implemente um programa de auxílio a uma eleição. Os votos válidos são representados pelos números 1, 2 e 3, cada um correspondendo a um candidato. O voto em branco é representado pelo número 0 e o voto nulo, pelo número -1. Esse fluxograma deverá processar N respostas da votação, as quais são lidas do teclado, separadas por espaço. O programa deve calcular e exibir:

- a) O total de votos para cada candidato
- b) O total de votos em branco
- c) O total de votos nulos
- d) O número do candidato vencedor (ou indicar se não houve vencedor, caso todos os votos tenham sido branco ou anulados)

## Problema 04:

Escreva e implemente uma solução que dado um número natural  $N$ , exiba a decomposição do mesmo em fatores primos, assim:

Exemplo:

$$6 = 2(1) 3(1)$$

$$9 = 3(2)$$

$$24 = 2(3) 3(1)$$

## Problema 05:

Escreva um programa que gere os  $N$  primeiros números perfeitos. Um número perfeito é aquele que é igual a soma dos seus divisores. Exemplo:  $6 = 1+2+3$