INSTITUTO FEDERAL DE MATO GROSSO
CAMPUS RONDONÓPOLIS
CURSO SUPERIOR DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Estrutura de Dados e Análise de Algoritmos

Prof. Daniel Domingos Alves

daniel.alves@ifmt.edu.br

Comandos de controle condicional e *Array* em C

31/10/2024

Comando if

- Em linguagem C, o comando **if** é utilizado quando for necessário escolher entre dois caminhos, ou quando se deseja executar um comando sujeito ao resultado de um teste.

Comando if

- A forma geral de um comando **if** é:

```
if (condição) {
    sequência de comandos;
}
```

- A expressão, na condição, será avaliada:
  - Se ela for zero (falsa), a declaração não será executada;
  - Se a condição for diferente de zero (verdadeira) a declaração será executada.

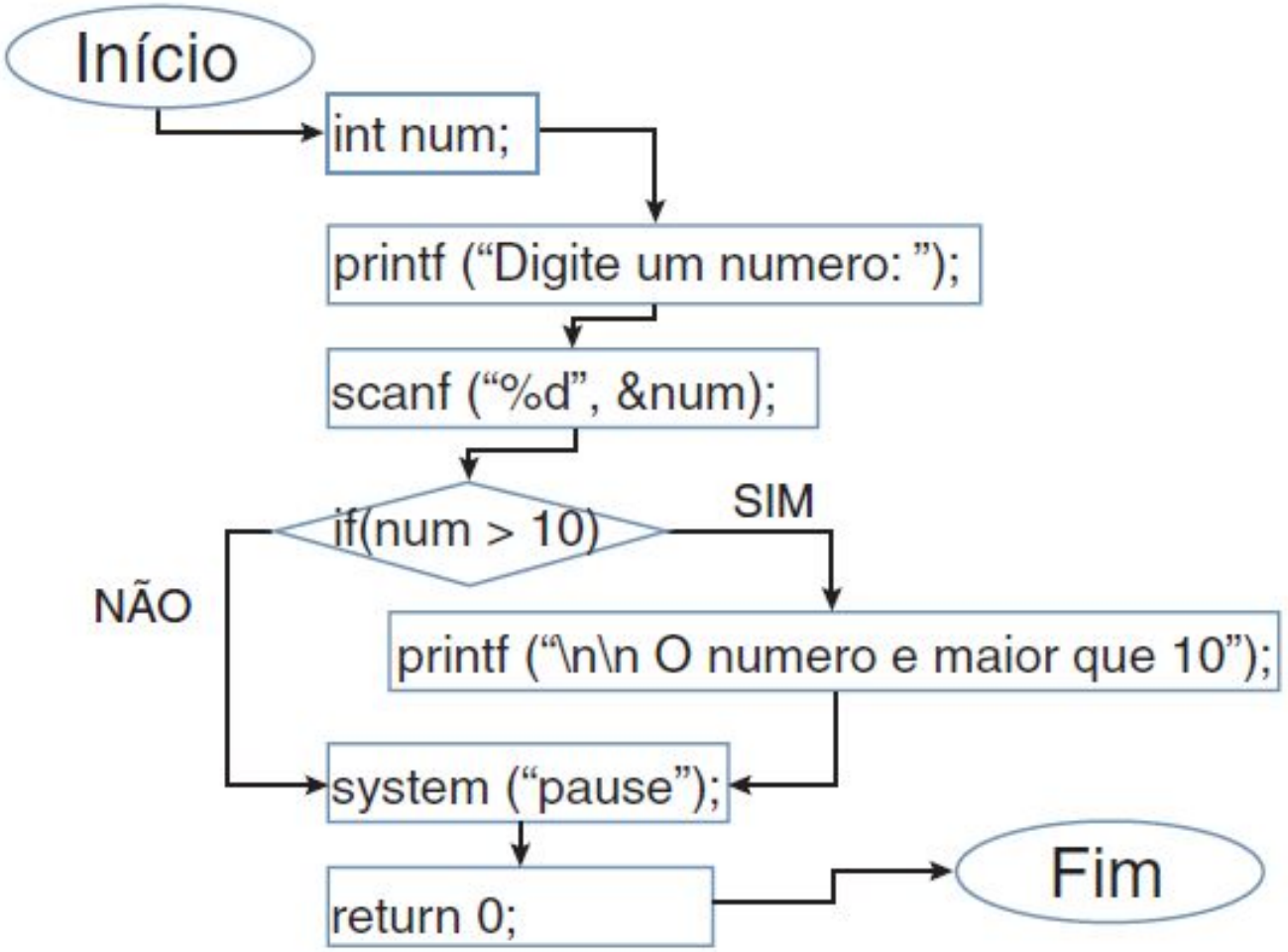
EXEMPLO IF

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);

    if(num > 10){
        printf("O numero eh maior do que 10.\n");
    }

    return 0;
}
```

EXEMPLO IF


Início
int num;
printf (“Digite um numero: ”);
scanf (“%d”, &num);
if(num > 10)
SIM
NÃO
printf (“\n\n O numero e maior que 10”);
system (“pause”);
return 0;
Fim

Condição do if

- A condição pode ser uma expressão usando operadores matemáticos, lógicos e relacionais
  - +,-, *, /, %
  - &&, ||
  - >, <, >=, <=, ==, !=
- Ex:
  - (x > 10 && y <= x-1)

COMANDO IF – USO DAS CHAVES { }

- Pode-se usar chaves { } para delimitar o bloco de instruções que pertence ao **if**

```
if (num > 10) {
    printf ("\n\n O numero eh maior que 10");
}
```

- As chaves devem ser usadas no caso de mais de uma instrução:

```
if (nota >= 60) {
    printf ("A nota é maior ou igual a 60 \n") ;
    printf ("O aluno está aprovado!") ;
}
```

- As chaves podem ser ignoradas se a instrução for única.

```
if (num > 10)
    printf ("\n\n O numero e maior que 10") ;
```

Exercício

- Dado o valor da nota de um aluno, monte a expressão **if** que verifica se ele precisará fazer a prova substitutiva. O aluno deverá fazer prova substitutiva se sua nota for maior ou igual a 30 e menor do que 60.

EXERCÍCIO

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num;
    printf ("Digite a nota: ");
    scanf ("%d",&num);
    if (num > 30  &&  num < 60)
        printf("O aluno deve fazer a prova sub \n");

    system("pause");
    return 0;
}
```

COMANDO ELSE

- O comando if-else tem a seguinte forma geral:

```
if(condição) {
     sequência de comandos 1;

} else{
     sequência de comandos 2;

}
```

Comando else

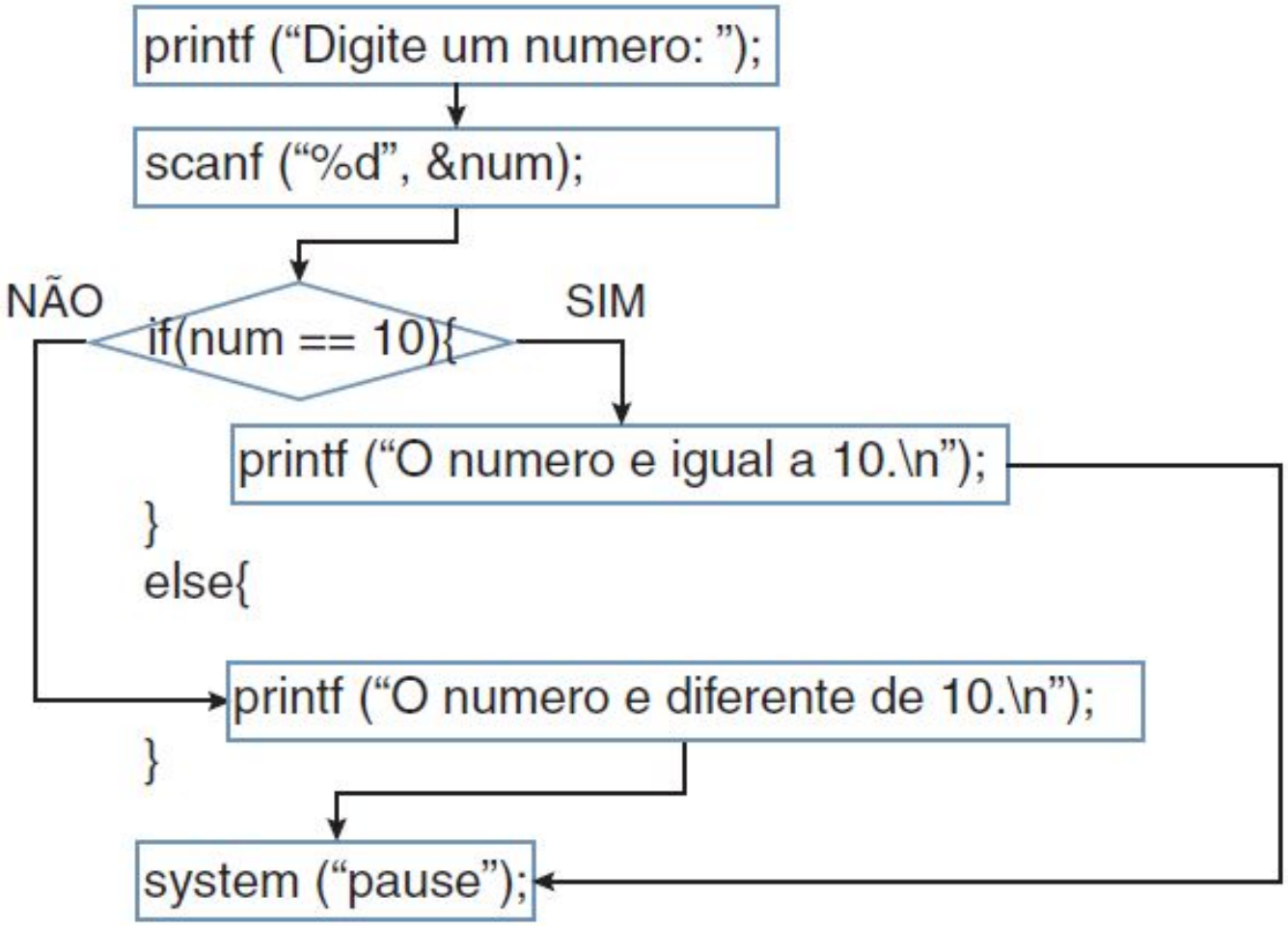
- A expressão da condição será avaliada:
  - Se ela for diferente de zero (verdadeiro), a sequência de comandos 1 será executada.
  - Se for zero (falso) a sequência de comandos 2 será executada.

- Note que quando usamos a estrutura if-else, uma das duas declarações será executada.
- Não há obrigatoriedade em usar o else

EXEMPLO IF-ELSE

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);
    if(num == 10){
        printf("O numero eh igual a 10.\n");
    }else{
        printf("O numero eh diferente de 10.\n");
    }
    return 0;
}
```

EXEMPLO IF-ELSE



COMANDO IF-ELSE

- Como no caso do comando if, as chaves podem ser ignoradas se a instrução contida no **else** for única.

```
if(num == 10){
    printf("O numero eh igual a 10.\n");
}else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

```
if(num == 10){
    printf("O numero eh igual a 10.\n");
}else{ // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

COMANDO IF-ELSE

- O comando do if é independe do comando do else

```
if(num == 10) //if sem usar chaves
    printf("O numero eh igual a 10.\n");
else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

```
if(num == 10){ //if com chaves
    printf("O numero eh igual a 10.\n");
}else // else sem usar chaves
    printf("O numero eh diferente de 10.\n");
```

```
if(num == 10){ //if com chaves
    printf("O numero eh igual a 10.\n");
}else{ // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

```
if(num == 10) //if sem usar chaves
    printf("O numero eh igual a 10.\n");
else{ // else com chaves
    printf("O numero eh diferente de 10.\n");
}
```

COMANDO IF-ELSE

Certo

```
if(condicao){
    sequência de comandos;
}else{
    sequência de comandos;
}
```

Errado

```
if(condicao){
    sequência de comandos;
else
    sequência de comandos;
}
```

A sequência de comandos de **if** é independente da sequência de comandos de **else**. Cada comando tem o seu próprio conjunto de chaves ({ }).

Aninhamento de if

- O **if** aninhado é simplesmente um **if** dentro da declaração de um outro **if** externo.
  - A estrutura if-else-if é apenas uma extensão da estrutura if-else.

- O único cuidado que devemos ter é o de saber exatamente a qual **if** um determinado **else** está ligado.

Aninhamento de if

```
if(condição){
    instrução 1;
    ...
    instrução N;
}else{
    if(condição){
        instrução 1;
        ...
        instrução N;
    }else{
        instrução 1;
        ...
        instrução N;
    }
}
```

```
if(condição){
    if(condição){
        instrução 1;
        ...
        instrução N;
    }else{
        instrução 1;
        ...
        instrução N;
    }
}else{
    instrução 1;
    ...
    instrução N;
}
```

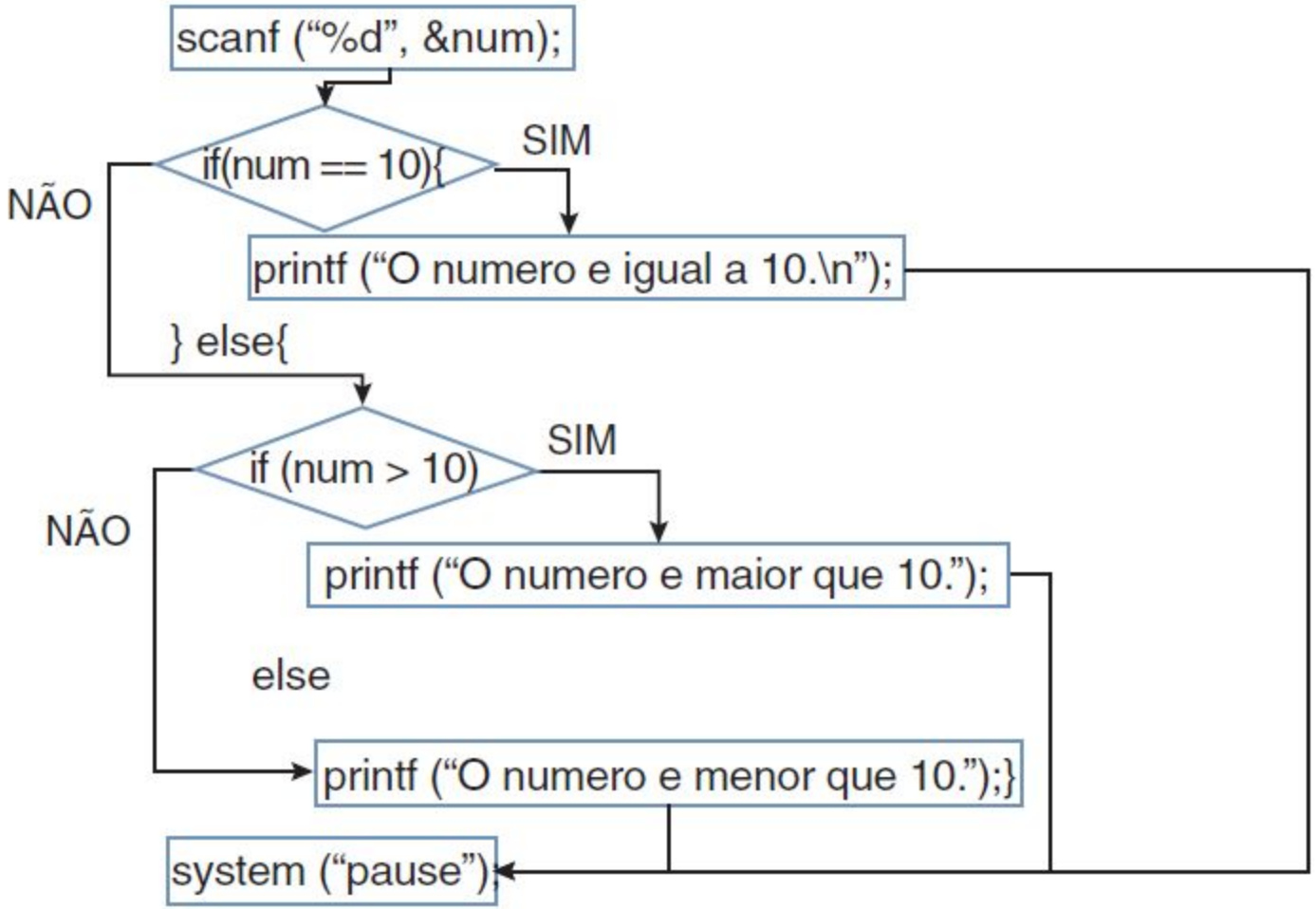
EXEMPLO ANINHAMENTO

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int num;
    printf("Digite um numero: ");
    scanf("%d",&num);

    if(num == 10){
        printf("O numero eh igual a 10.\n");
    }else{
        if(num > 10)
            printf("O numero eh maior do que 10.\n");
        else
            printf("O numero eh menor do que 10.\n");
    }

    return 0;
}
```

EXEMPLO ANINHAMENTO



Exercício

- Dado o valor da nota de um aluno, monte o conjunto de **if's** e **else's** que verifica se foi aprovado, reprovado ou precisará fazer a prova substitutiva.

EXERCÍCIO

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int d;
    printf("Digite a nota: ");
    scanf("%d",&d);
    if (d >= 60)
        printf("Aluno aprovado \n");
    else
        if(d < 30)
            printf("Aluno reprovado \n");
        else
            printf("O aluno deve fazer a prova sub \n");


    return 0;
}
```

EXERCÍCIO

- Construir a sequência de if-else para escrever o nome do dígito lido
  - ‘0’ -> “zero”;
  - ‘1’ -> “um”;
  - etc.

EXERCÍCIO

- Construir a seqüência de if-else para escrever o nome do dígito lido
  - ‘0’ -> “zero”;
  - ‘1’ -> “um”;
  - etc.

```
char ch;
scanf("%c",&ch);
if (ch == '0') printf("Zero");
else if (ch=='1') printf("Um");
else if (ch=='2') printf("Dois");
else if ...
else if (ch=='9') printf("Nove");
else printf("Nao era um digito!");
```

Expressão Condicional

- Quando o compilador avalia uma condição, ele quer um valor de retorno para poder tomar a decisão.
- Esta expressão não necessita ser uma expressão no sentido convencional.
- Uma variável sozinha pode ser uma "expressão" e esta retornar o seu próprio valor.

O Operador ?

- Também conhecido como operador ternário
- A expressão condicional “**? :**” é uma simplificação do if-else utilizada tipicamente para atribuições condicionais

Sintaxe:

Condição ? verdadeiro : falso

Onde:

- **Condição** é a condição que será testada.
- **Verdadeiro** é o que fazer quando a condição for verdadeira.
- **Falso** é o que fazer quando a condição for falsa.

O Operador ?

- Uma expressão como

```
if (a > 0)
    b = -150;
else
    b = 150;
```

- pode ser simplificada usando-se o operador ? da seguinte maneira:

```
b = a > 0 ? -150 : 150;
```

Exercício

- Dado dois números x e y, retorne o maior na variável z:
  - Usando if-else
  - Usando o operador ternário

Exercício

Usando if-else

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int x,y,z;
    printf("Digite x:");
    scanf("%d",&x);
    printf("Digite y:");
    scanf("%d",&y);
    if(x > y)
        z = x;
    else
        z = y;
    printf("Maior = %d\n",z);

    return 0;
}
```

Usando operador ternário

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int x,y,z;
    printf("Digite x:");
    scanf("%d",&x);
    printf("Digite y:");
    scanf("%d",&y);
    z = x > y ? x : y;
    printf("Maior = %d\n",z);

    return 0;
}
```

O COMANDO SWITCH

- O comando switch é próprio para se testar uma variável em relação a diversos valores pré-estabelecidos.
  - Parecido com if-else-if, porém não aceita expressões, **apenas constantes**.
  - O switch testa a variável e executa a declaração cujo “case” corresponde ao valor atual da variável.

O COMANDO SWITCH

- Forma geral do comando switch

```
switch (expressão) {
    case valor 1:
        sequência de comandos 1;
        break;
    case valor k:
        sequência de comandos k;
        break;
    ...
    default:
        sequência de comandos padrão;
        break;
}
```

O COMANDO SWITCH

- O comando switch
  - Avalia o valor da **expressão** com os valores associados às cláusulas **case** em sequência;
  - Quando o valor associado a uma cláusula é igual ao valor da **expressão,** os respectivos comandos são executados até encontrar um **break.**
- A declaração **default** é opcional e será executada apenas se a **expressão** que está sendo testada não for igual a nenhuma das constantes presentes nos **case**.

O COMANDO SWITCH

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    char ch;
    printf("Digite um simbolo de pontuacao: ");
    ch = getchar();
    switch( ch ) {
        case '.':
            printf("Ponto.\n"); break;
        case ',':
            printf("Virgula.\n"); break;
        case ':':
            printf("Dois pontos.\n"); break;
        case ';':
            printf("Ponto e virgula.\n"); break;
        default :
            printf("Nao eh pontuacao.\n");
    }

    return 0;
}
```

O COMANDO SWITCH

- O comando break
  - Faz com que o switch seja interrompido assim que uma das sequências de comandos seja executada.
  - Não é essencial. Se após a execução da declaração não houver um break, o programa continuará executando o próximo comando case.
  - Isto pode ser útil em algumas situações, mas tenha cuidado.

O COMANDO SWITCH SEM BREAK

```
int num;
scanf("%d",&num);
switch( num ) {
    case 0: printf("0"); /* 0123456789 */
    case 1: printf("1"); /* 123456789 */
    case 2: printf("2"); /* 23456789 */
    case 3: printf("3"); /* 3456789 */
    case 4: printf("4"); /* 456789 */
    case 5: printf("5"); /* 56789 */
    case 6: printf("6"); /* 6789 */
    case 7: printf("7"); /* 789 */
    case 8: printf("8"); /* 89 */
    case 9: printf("9"); /* 9 */
}
```

EXERCÍCIO

- Construir o switch para escrever o nome do dígito lido
  - 0 -> “zero”;
  - 1 -> “um”;
  - etc.

EXERCÍCIO

- Construir o switch para escrever o nome do dígito lido
  - 0 -> “zero”;
  - 1 -> “um”;
  - etc.

```
switch(num) {
    case 0: printf("Zero"); break;
    case 1: printf("Um"); break;
    case 2: printf("Dois"); break;
    case 3: printf("Tres"); break;
    case 4: printf("Quatro"); break;
    case 5: printf("Cinco"); break;
    case 6: printf("Seis"); break;
    case 7: printf("Sete"); break;
    case 8: printf("Oito"); break;
    case 9: printf("Nove"); break;
}
```

Array

- Array ou “vetor” é a forma mais familiar de dados estruturados.
- Basicamente, um array é uma sequência de elementos do mesmo tipo, onde cada elemento é identificado por um índice
  - A ideia de um array ou “vetor” é bastante simples: criar um conjunto de variáveis do mesmo tipo utilizando apenas um nome.

Array - Problema

- Imagine o seguinte problema
  - leia as notas de uma turma de cinco estudantes e depois imprima as notas que são maiores do que a média da turma.
- Um algoritmo para esse problema poderia ser o mostrado a seguir.

ARRAY - SOLUÇÃO

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  float n1,n2,n3,n4,n5;
  printf("Digite a nota de 5 estudantes: ");
  scanf("%f",&n1);
  scanf("%f",&n2);
  scanf("%f",&n3);
  scanf("%f",&n4);
  scanf("%f",&n5);
  float media = (n1+n2+n3+n4+n5)/5.0;
  if(n1 > media) printf("nota: %f\n",n1);
  if(n2 > media) printf("nota: %f\n",n2);
  if(n3 > media) printf("nota: %f\n",n3);
  if(n4 > media) printf("nota: %f\n",n4);
  if(n5 > media) printf("nota: %f\n",n5);

  return 0;
}
```

Array

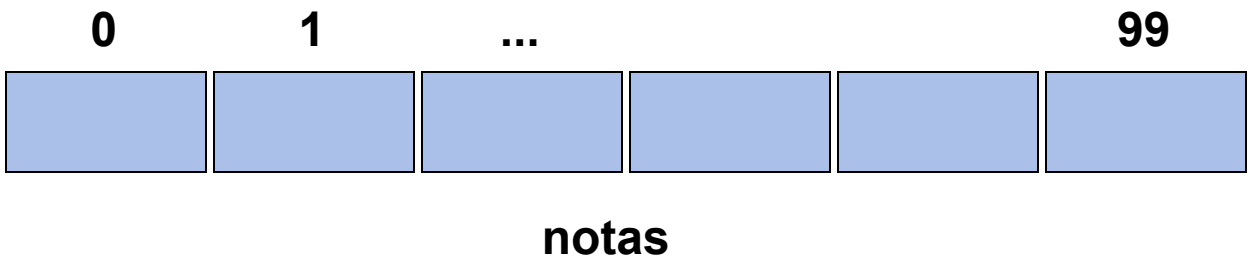
- O algoritmo anterior apresenta uma solução possível para o problema apresentado
- Porém, essa solução é inviável para grandes quantidades de alunos
  - Imagine se tivéssemos de processar as notas de 100 alunos

ARRAY

- Para 100 alunos, precisamos de:
  - Uma variável para armazenar a nota de cada aluno
    - **100 variáveis**
  - Um comando de leitura para cada nota
    - **100 scanf()**
  - Um somatório de **100 notas**
  - Um comando de teste para cada aluno
    - **100 comandos if.**
  - Um comando de impressão na tela para cada aluno
    - **100 printf()**

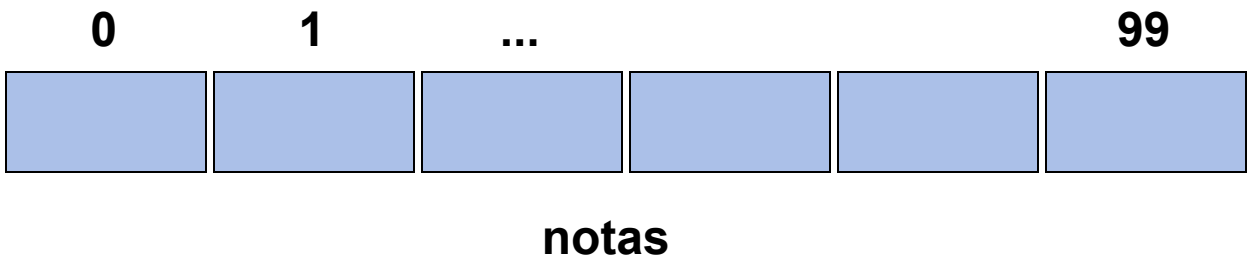
Array - Definição

- As variáveis têm relação entre si
  - todas armazenam notas de alunos
- Podemos declará-las usando um ÚNICO nome para todos os 100 alunos
  - notas: conjunto de 100 valores acessados por um índice
  - Isso é um **array**!



Array - Declaração

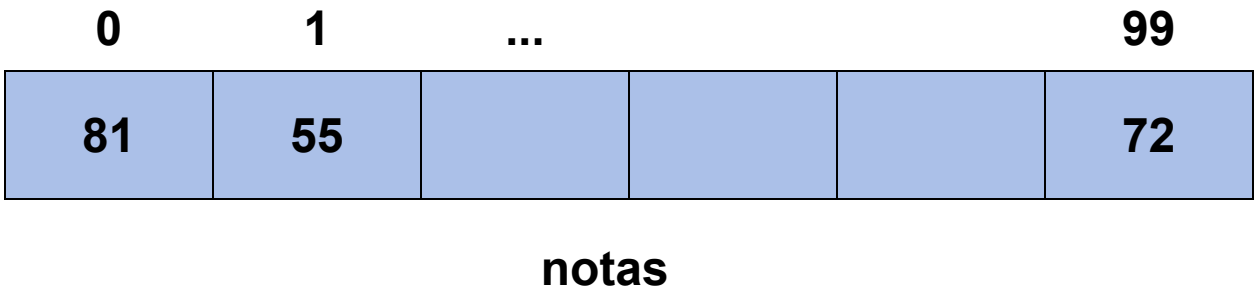
- Arrays são agrupamentos de dados adjacentes na memória. Declaração:
  - *tipo_dado nome_array[tamanho];*
- O comando acima define um array de nome **nome_array,** capaz de armazenar **tamanho** elementos adjacentes na memória do tipo **tipo_dado**
  - Ex: **int notas[100];**



Array - Declaração

- Em um array, os elementos são acessados especificando o índice desejado entre **colchetes [ ]**
- A numeração começa sempre do zero
- Isto significa que um array de 100 elementos terá índices de 0 a 99:
  - notas[0], notas[1], notas[2], ..., notas[99]

```
int notas[100];
notas[0] = 81;
notas[1] = 55;
...
notas[99] = 72;
```



Array - Definição

- Observação
  - Se o usuário digitar mais de 100 elementos em um array de 100 elementos, o programa tentará ler normalmente.
  - Porém, o programa os armazenará em uma parte não reservada de memória, pois o espaço reservado para o array foi para somente 100 elementos.
  - Isto pode resultar nos mais variados erros durante a execução do programa.

ARRAY = VARÍAVEL

- Cada elemento do array tem todas as características de uma variável e pode aparecer em expressões e atribuições (respeitando os seus tipos)
  - notas[2] = x + notas[3];
  - if (notas[2] > 60)
- Ex: somar todos os elementos de notas:

```
int soma = 0;
for(i=0;i < 100; i++)
    soma = soma + notas[i];
```

Percorrendo um array

- Podemos usar um comando de repetição (for, while e do-while) para percorrer um array
- Exemplo: somando os elementos de um array de 5 elementos

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    int lista[5] = {3,51,18,2,45};
    int i, soma = 0;
    for(i = 0; i < 5; i++)
        soma = soma + lista[i];

    printf("soma = %d\n",soma);

    return 0;
}
```

Variáveis		
soma	i	lista[i]
0		
3	0	3
54	1	51
72	2	18
74	3	2
119	4	45
	5	

Array - Características

- Características básicas de um Array
  - 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

Array - Problema

- Voltando ao problema anterior
  - leia as notas de uma turma de cinco estudantes e depois imprima as notas que são maiores do que a média da turma.

ARRAY - SOLUÇÃO

- Um algoritmo para esse problema usando array:

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    float notas[5];
    int i;
    printf("Digite as notas dos estudantes\n");
    for(i = 0; i < 5; i++){
        printf("Nota do estudante %d:",i);
        scanf("%f",&notas[i]);
    }
    float media = 0;
    for(i = 0; i < 5; i++)
        media = media + notas[i];
    media = media / 5;

    for(i = 0; i < 5; i++)
        if(notas[i] > media)
            printf("Notas: %f\n",notas[i]);

    return 0;
}
```

ARRAY - SOLUÇÃO

- Se ao invés de 5, fossem 100 alunos?

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
    float notas[100];
    int i;
    printf("Digite as notas dos estudantes\n");
    for(i = 0; i < 100; i++){
        printf("Nota do estudante %d:",i);
        scanf("%f",&notas[i]);
    }
    float media = 0;
    for(i = 0; i < 100; i++)
        media = media + notas[i];
    media = media / 100;

    for(i = 0; i < 100; i++)
        if(notas[i] > media)
            printf("Notas: %f\n",notas[i]);

    return 0;
}
```

Exercício

- Para um array A com 5 números inteiros, formular um algoritmo que determine o maior elemento deste array

Exercício - Solução

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  int i, A[5] = {3,18,2,51,45};
  int ma = A[0];

  for(i=1; i<5; i++){
    if(ma < A[i])
        ma = A[i];
  }

  printf("Maior = %d\n", ma);

  return 0;
}
```

Variáveis		
ma	i	A[i]
3	0	3
18	1	18
51	2	2
	3	51
	4	45
	5	

Copiando um array

- Não se pode fazer atribuição de arrays inteiros, apenas de suas posições individualmente

```c
#include <stdio.h>
#include <stdlib.h>
int main(){
  int v[5] = {1,2,3,4,5};
  int v1[5];

  v1 = v; //ERRADO!

  int i;
  for(i=0; i<5; i++)
    v1[i] = v[i]; //CORRETO

  return 0;
}
```