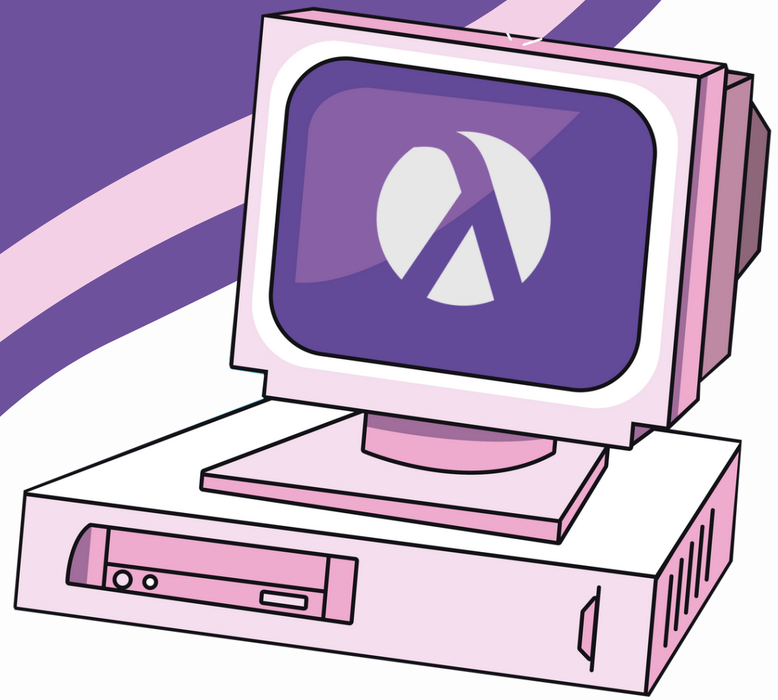
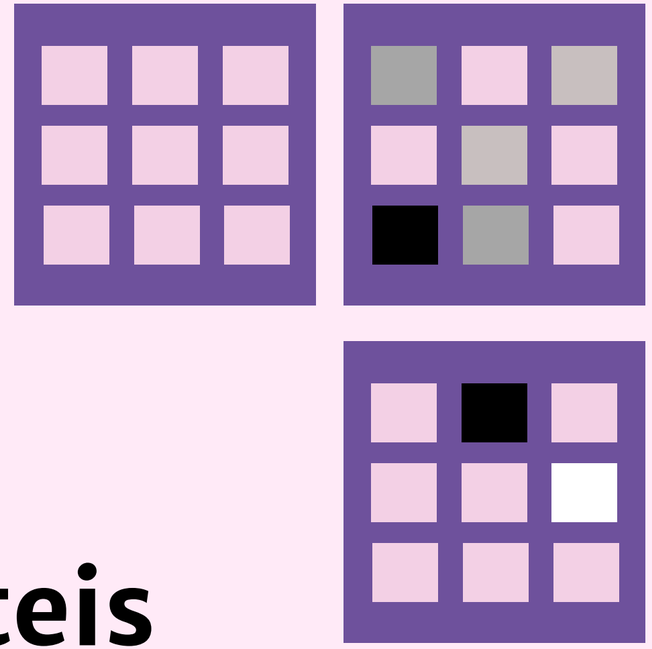
LÓGICA DE PROGRAMAÇÃO E C

DIA 3



VETORES

VETORES

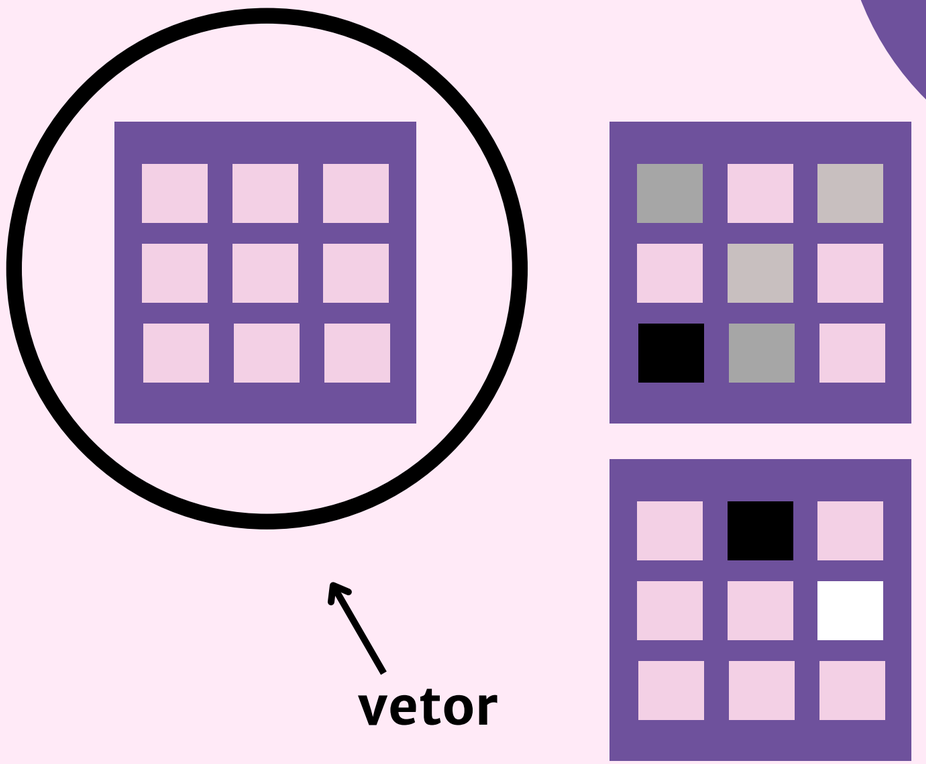


"virou física isso aqui?"

- O que são?
- O porquê de eles serem úteis
- Como usá-los?

O que são?

"O vetor é uma estrutura de dados indexada, que pode armazenar uma determinada quantidade de valores do mesmo tipo."

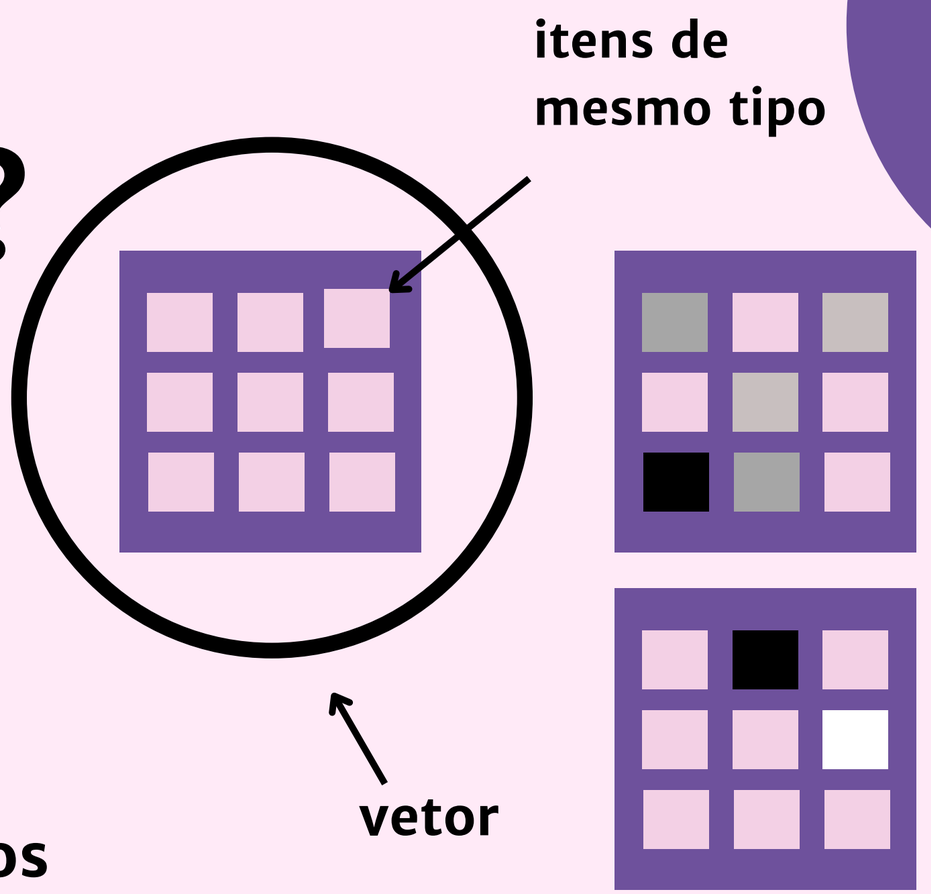


- **Ou seja, é como ter uma caixa para guardar itens do mesmo tipo.**

Pode ser uma caixa de grampos, uma caixa de bombons, ou, em C, uma "caixa" de caracteres, por exemplo.

Por que utilizá-los?

Você resume uma grande quantidade de variáveis (seus dados) a uma única a qual contém todos reunidos.

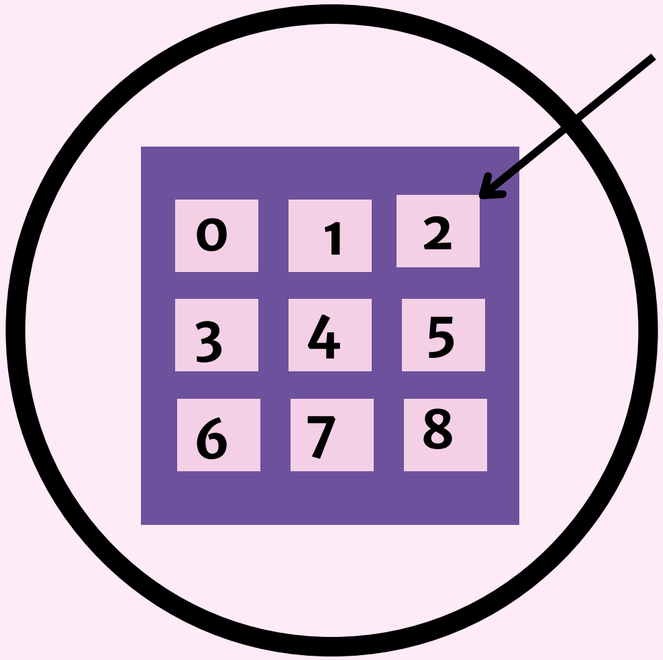


- **É melhor carregar uma caixa de grampos cheia do que carregar, um por um, cada grampo desta caixa.**

Ao trabalhar em um programa, é melhor referenciar uma única variável com 50 itens do que os 50 itens, um por vez.

Como utilizá-los?

Tipo NomeDoVetor[quantia_de_elementos]



NOTE: Diferente no nosso convencional, a contagem dos elementos de um vetor na linguagem C começa pelo ZERO!

Imagine que precisemos armazenar vinte valores inteiros, em ordem, para utilizarmos em nosso código

Esse seria nosso código sem o uso de vetores

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

int main (){

    printf("Digite um codigo: ");
    int a;
    scanf("%d", &a);

    printf("Digite um codigo: ");
    int b;
    scanf("%d", &b);

    printf("Digite um codigo: ");
    int c;
    scanf("%d", &c);

    printf("Digite um codigo: ");
    int d;
    scanf("%d", &d);

    printf("Digite um codigo: ");
    int e;
    scanf("%d", &e);

    printf("Digite um codigo: ");
    int f;
    scanf("%d", &f);
```

```c
    printf("Digite um codigo: ");
    int g;
    scanf("%d", &g);

    printf("Digite um codigo: ");
    int h;
    scanf("%d", &h);

    printf("Digite um codigo: ");
    int i;
    scanf("%d", &i);

    printf("Digite um codigo: ");
    int j;
    scanf("%d", &j);

    printf("Digite um codigo: ");
    int k;
    scanf("%d", &k);

    printf("Digite um codigo: ");
    int l;
    scanf("%d", &l);

    printf("Digite um codigo: ");
    int m;
    scanf("%d", &m);

    printf("Digite um codigo: ");
    int n;
    scanf("%d", &n);
```

```c
    printf("Digite um codigo: ");
    int o;
    scanf("%d", &o);

    printf("Digite um codigo: ");
    int p;
    scanf("%d", &p);

    printf("Digite um codigo: ");
    int q;
    scanf("%d", &q);

    printf("Digite um codigo: ");
    int r;
    scanf("%d", &r);

    printf("Digite um codigo: ");
    int s;
    scanf("%d", &s);

    printf("Digite um codigo: ");
    int t;
    scanf("%d", &t);
```

Utilizando vetores:

```c
int ValoresOrdenados[20];

for (int i = 0; i < 20; i++) {

    printf("Digite um valor: ");
    scanf("%d", &ValoresOrdenados[i]);
}
```

Pronto! Agora temos os mesmos vinte valores armazenados, mas de maneira bem mais simplificada e fácil de utilizar ao longo dos nossos programas

Agora, imaginemos um caso em que precisamos mostrar na tela os inteiros salvos:

```
printf("\nnumero: %d", a);
printf("\nnumero: %d", b);
printf("\nnumero: %d", c);
printf("\nnumero: %d", d);
printf("\nnumero: %d", e);
printf("\nnumero: %d", f);
```

o mesmo processo aconteceria para os vinte valores (não anexado aqui para poupar a nossa paciência, mas deu para pegar a ideia, certo?)

versão com vetores

```
for (int i = 0; i < 20; i++) {
    printf("\nnumero da posicao %d: %d", (i+1), ValoresOrdenados[i]);
}
```

Para fazer uma soma, é mais fácil assim?

```
int soma = a + b + c + d + e + f + g + h + i + j + k + l + m + n + o + p + q + r + s + t;
printf("Resultado da soma: %d\n", soma);
```

Ou assim?

```
int soma = 0;
for (int i = 0; i < 20; i++) {
    soma = soma + ValoresOrdenados[i];
}
printf("Resultado da soma: %d\n", soma);
```

- **Apesar do primeiro código parecer mais claro, mais simples de ler e entender, é menos prático, pois te faz somar "manualmente" cada item do seu vetor.**

- **Apesar do primeiro código parecer mais claro, mais simples de ler e entender, é menos prático, pois te faz somar "manualmente" cada item do seu vetor.**
- **Imagine armazenar mil números.**

- Apesar do primeiro código parecer mais claro, mais simples de ler e entender, é menos prático, pois te faz somar "manualmente" cada item do seu vetor.
- Imagine armazenar mil números.
- Você teria que digitar, uma por uma, cada variável.

- Apesar do primeiro código parecer mais claro, mais simples de ler e entender, é menos prático, pois te faz somar "manualmente" cada item do seu vetor.
- Imagine armazenar mil números.
- Você teria que digitar, uma por uma, cada variável.
- Agora imagine cem mil.

- Apesar do primeiro código parecer mais claro, mais simples de ler e entender, é menos prático, pois te faz somar "manualmente" cada item do seu vetor.
- Imagine armazenar mil números.
- Você teria que digitar, uma por uma, cada variável.
- Agora imagine cem mil.
- Ou ainda, um milhão!

Exercícios

Exercícios

1. Projete um programa que some todos os valores de um vetor de 9 números.

Exercícios

1. Projete um programa que some todos os valores de um vetor de 9 números.

2. Projete um programa que encontre o valor máximo de um vetor com 6 números.

Exercícios

1. Projete um programa que some todos os valores de um vetor de 9 números.

2. Projete um programa que encontre o valor máximo de um vetor com 6 números.

3. Projete um programa que verifique se um vetor com 12 números inteiros positivos tem mais números pares ou ímpares.

Exercícios

4. A Láurea Acadêmica é uma homenagem prestada a alunos que tiveram elevado nível de aproveitamento no curso de graduação. Na UEM, todos os alunos que tiveram mais do que 2/3 das notas finais das disciplinas maiores ou iguais a 9,0 recebem esta homenagem. Projete um programa que receba as notas finais de um aluno e determine se ele receberá a Láurea Acadêmica.

Exercícios

4. A Láurea Acadêmica é uma homenagem prestada a alunos que tiveram elevado nível de aproveitamento no curso de graduação. Na UEM, todos os alunos que tiveram mais do que 2/3 das notas finais das disciplinas maiores ou iguais a 9,0 recebem esta homenagem. Projete um programa que receba as notas finais de um aluno e determine se ele receberá a Láurea Acadêmica.

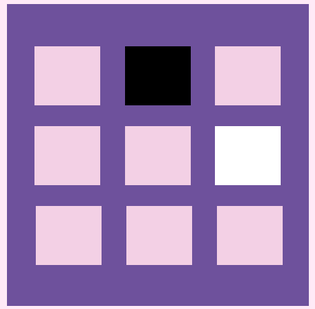
5. Projete um programa que encontre o índice (posição) da primeira ocorrência do valor máximo de um vetor não vazio

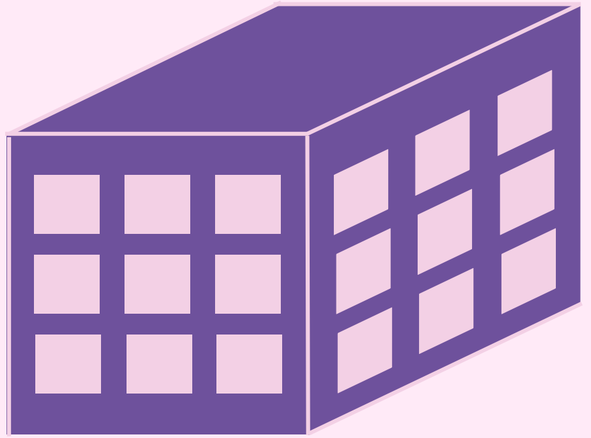
MATRIZES

MATRIZES

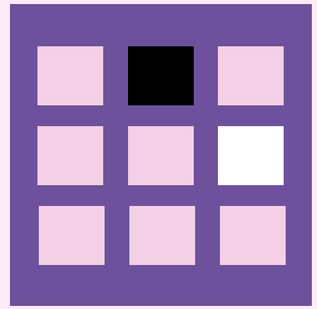
"agora virou matemática??"

- O que são, onde vivem, do que se alimentam?
- Como declará-las?
- Operações com matrizes





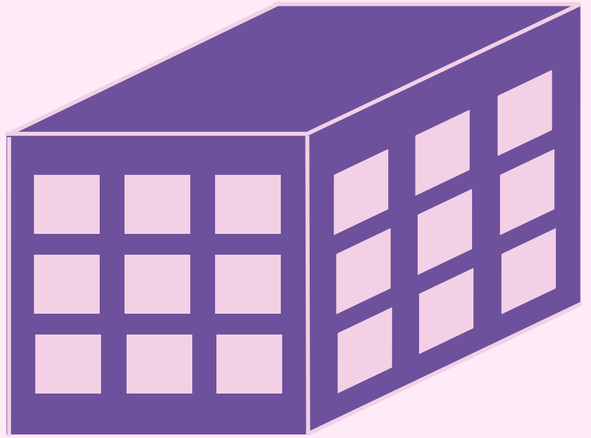
O que são, onde vivem, do que se alimentam?



"Assim como os vetores, as 'matrizes' são uma estrutura de dados indexada, porém são multidimensionais [...]

- **Ou seja, é como ter uma caixa com várias divisões para guardar itens de vários tipo.**

[...] e também como os vetores, você resume uma grande quantidade de variáveis (seus dados) a uma única a qual contém todos reunidos."



Como declara-las?

Tipo NomeDoVetor[x][y][z]

- **Tipo** — tipo da variável
- **NomeDoVetor** — nome da variável
- **[x][y][z]** — tamanho das dimensões

Exemplos:

```
int array2D[5][4];
```

```
int x = 20;
int y = 10;
int A[x][15][10];
```

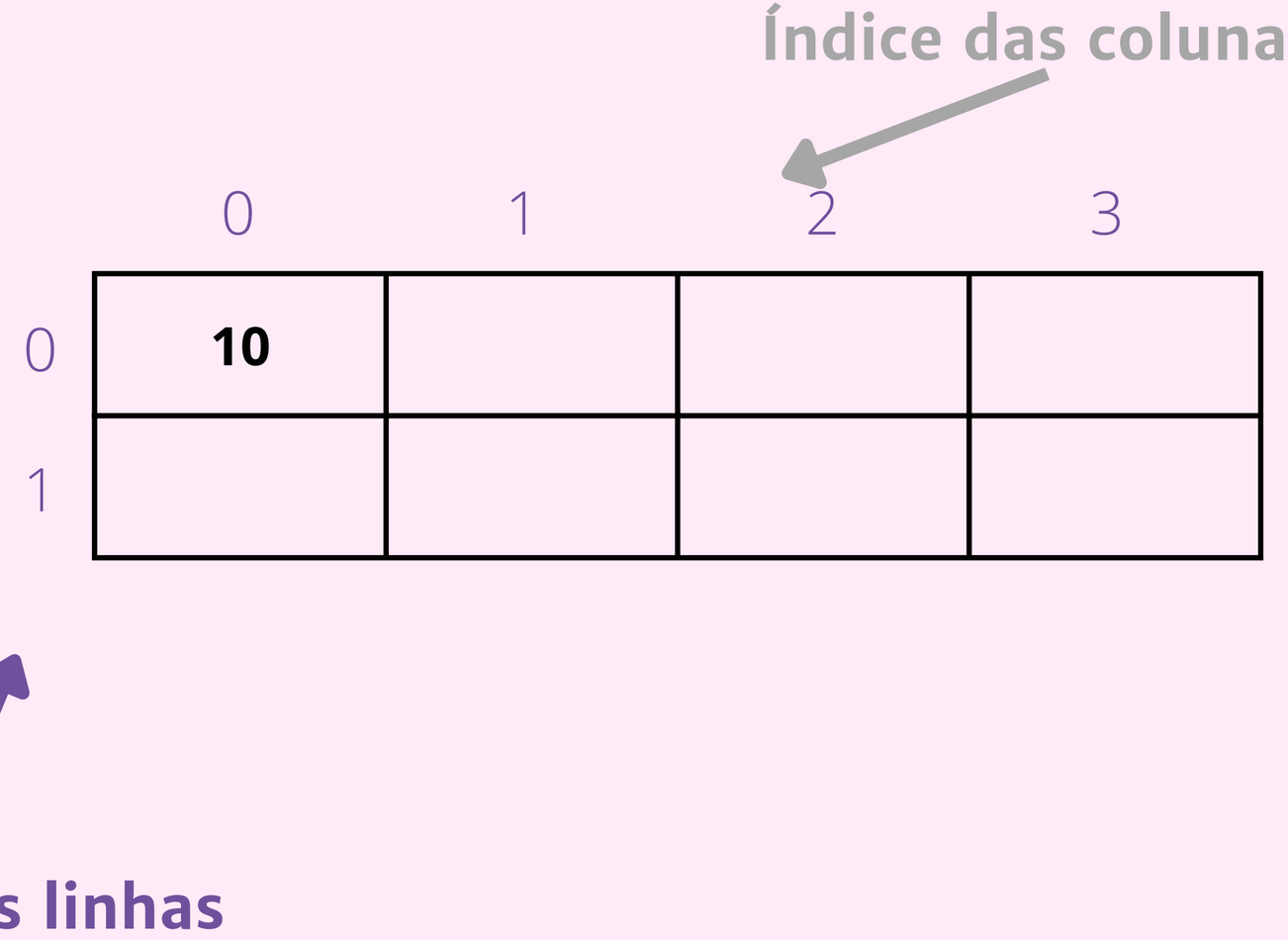
Como declara-las com valores?

```
int NomeDoVetor[3][4] = {{45, 75, 53}, {10, -50, 40}};
```

	0	1	2	3
0	**45**	**75**	**53**	
1	**10**	**-50**	**40**	
2				

Como armazenar os valores?

```
int A[2][4];
A[0][0] = 10;
```



Ou:

```
int A[2][4];

for(int i=0; i<2; i++){
    for(int j=0; i<4; j++){
        A[i][j]=10;
    }
}
```

	0	1	2	3
0	**10**	**10**	**10**	**10**
1	**10**	**10**	**10**	**10**

Operações com matrizes:

Podemos operar as matrizes normalmente, igual fazemos com as variáveis unidimensionais, exemplos:

```
float A[3][3] = {{45.5, 75, 53}, {10, -5.6, 40}, {9, -4, 0}};

A[0][0] == 45.5; //true
A[2][1] == -4; //true
A[0][0] + A[2][1] == 41.5; //true
A[0][0] != 45.5; //false
```

Exercícios

Exercícios

1. Projete um programa que receba todos os valores de uma matriz 2x2 e que soma e printa todos esses valores.

Exercícios

1. Projete um programa que receba todos os valores de uma matriz 2x2 e que soma e printa todos esses valores.

2. faça 3 series de 15 agachamentos, 10 flexões e 10 abdominais.

Exercícios

1. Projete um programa que receba todos os valores de uma matriz 2x2 e que soma e printa todos esses valores.

2. faça 3 series de 15 agachamentos, 10 flexões e 10 abdominais.

3. Projete um programa que calcula a soma da diagonal principal de uma matriz quadrada qualquer (o programa deve perguntar a dimensão e os valores da matriz).

Exercícios

1. Projete um programa que receba todos os valores de uma matriz 2x2 e que soma e printa todos esses valores.

2. faça 3 series de 15 agachamentos, 10 flexões e 10 abdominais.

3. Projete um programa que calcula a soma da diagonal principal de uma matriz quadrada qualquer (o programa deve perguntar a dimensão e os valores da matriz).

4. Projete um programa que receba todos os valores de uma matriz 3x3 e que armazene e printe a transposta dessa matriz.

STRUCTURES

STRUCTURES

"Meteu um Inglês?"

- O que são?
- Utilidade
- Como declarar e usar?

O que são?

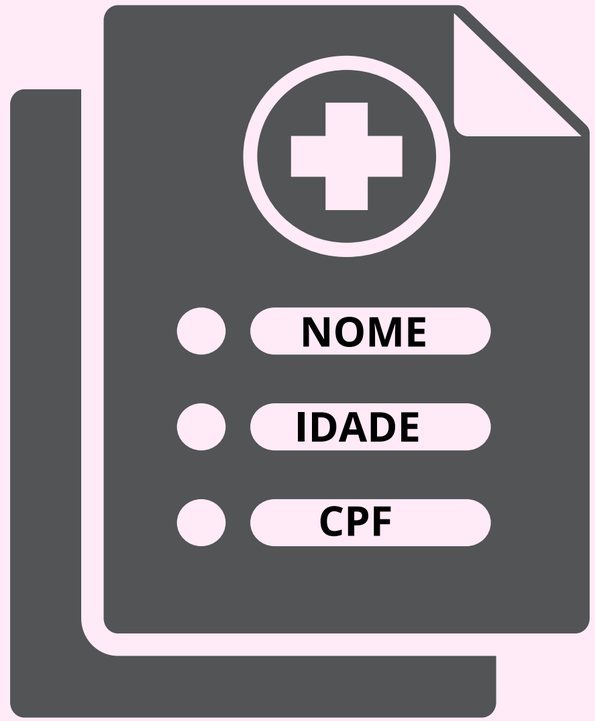
Uma structure é uma variável heterogênea que armazena um conjunto de dados logicamente relacionados.

- **De forma mais simples, é uma ficha.**

Podemos visualizar como uma ficha de hospital, ou em C, uma "ficha" de dados.

Utilidade

Facilita a visualização e manipulação de dados os quais estão ligados entre si.

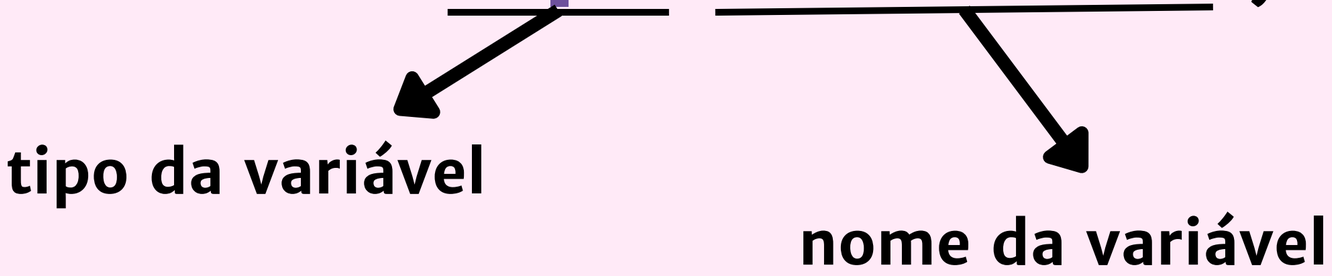


- **É mais fácil acessar dados quando eles estão seguindo um padrão.**

Quando tratamos de um programa, é mais eficiente manter os dados organizados, facilitando o acesso e manipulação de dados.

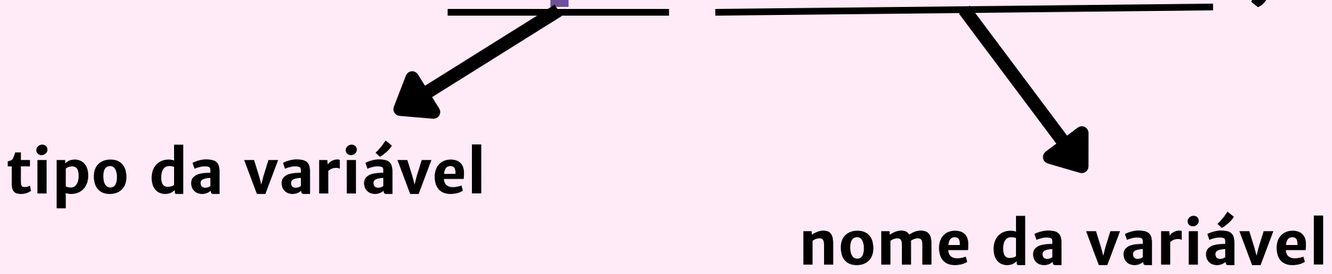
Como declará-las? Forma (1)

```
typedef struct {

    Tipo NomeVar1;

} NomeDaStruct;
```



Também é possível declarar structs desse jeito. Forma (2)

```
struct NomeDaStruct{

    Tipo NomeVar1;

};
```



Exemplos:

Consideremos que iremos armazenar alguns dados sobre pessoas diversas

Sem utilizar struct, teríamos algo assim:

```c
int main(){

float JoãoPeso;
float PedroPeso;
float CaioPeso;
float VassaloPeso;
int JoãoIdade;
int PedroIdade;
int CaioIdade;
int VassaloIdade;
float JoãoAltura;
float PedroAltura;
float CaioAltura;
float VassaloAltura;
}
```

```c
int main(){

char Pessoa1Nome[30];
char Pessoa1Tpsanguineo[3];
char Pessoa2Nome[30];
char Pessoa2Tpsanguineo[3];
char Pessoa3Nome[30];
char Pessoa3Tpsanguineo[3];
char Pessoa4Nome[30];
char Pessoa4Tpsanguineo[3];
char Pessoa5Nome[30];
char Pessoa5Tpsanguineo[3];
char Pessoa6Nome[30];
char Pessoa6Tpsanguineo[3];
}
```

Utilizando structures:

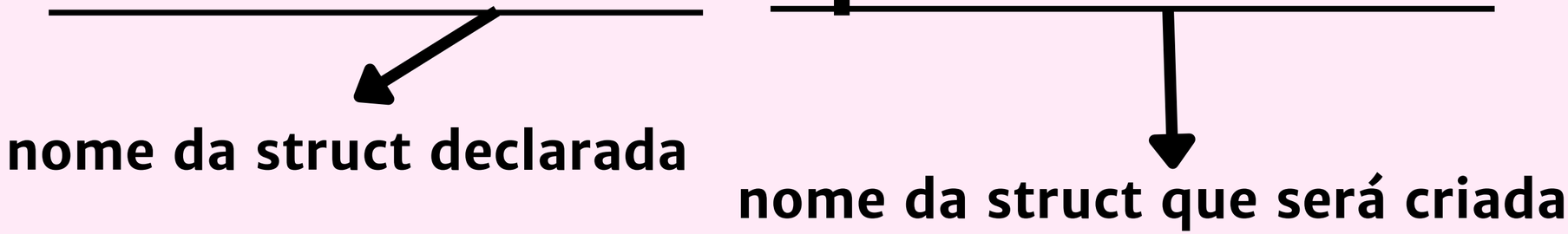
```c
typedef struct{
    float Peso;
    int Idade;
    float Altura;
    char Tpsanguineo[3];
} Pessoa;

int main(){

}
```

```c
struct Pessoas{
    float Peso;
    int Idade;
    float Altura;
    char Tpsanguineo[3];
};

int main(){

}
```

Podemos fazer de forma simples e antes do PROGRAMA PRINCIPAL, sem precisar necessariamente saber a quantidade de dados que serão armazenados

Como criar e acessar um dado em struct? Para a Forma (1)

Criar: NomeDaStruct ApelidoDaStruct



Acessar: ApelidoDaStruct.CampoDaStruct

Mas para a Forma (2) temos:

Criar: `struct NomeDaStruct ApelidoDaStruct`

- `NomeDaStruct`: nome da struct declarada
- `ApelidoDaStruct`: nome da struct que será criada

Acessar: `ApelidoDaStruct.CampoDaStruct`

- `ApelidoDaStruct`: nome da struct criada
- `CampoDaStruct`: nome do campo da struct declarada

Exemplos:

```c
int main(){

Pessoa Joao;

Joao.Peso = 45.5;
scanf("%d", &Joao.Idade);
Joao.Altura = 1.51;

}
```

```c
int main(){

struct Pessoa Pablo;

Pablo.Cintura = 69.3;
scanf("%d", &Pablo.Peito);
Pablo.Panturrilha = 33.3;

}
```

Operações com structures:

Como os campos de uma struct são variáveis, as operações básicas de interação entre variáveis podem ser utilizadas adicionando apenas a operação de atribuição de todos os campos entre as structs

```
int main(){

struct Pessoa Joao, Pablo;

Joao.Peso = 40.5; Joao.Idade = 18; Joao.Altura = 1.51;

Pablo = Joao; // Pablo.Peso == 40.5; Pablo.Idade == 18; Pablo.Altura == 1.51; (true)
}
```

Também podemos fazer uma junção entre structs e vetores:

Assim generalizamos a utilização de structs

```c
int main() {
    struct Pessoa Participante[10];

    Participante[0].Peso = 32.5;
    Participante[0].Idade = 10;

    Participante[1].Peso = 45.3;
    scanf("%d", &Participante[1].Idade);

    scanf("%f", &Participante[2].Peso);
    scanf("%d", &Participante[2].Idade);


}
```

```c
int main() {
    struct Pessoa Participante[10];

    for (int i = 0; i < 10; i++) {
        scanf("%f", &Participante[i].Peso);
        scanf("%d", &Participante[i].Idade);
        scanf("%f", &Participante[i].Altura);
    }

}
```

Exercícios

Exercícios

1. Utilizando structures, projete um programa que receba nome, idade e o peso de 5 alunos e imprima esses dados na tela

Exercícios

1. Utilizando structures, projete um programa que receba nome, idade e o peso de 5 alunos e imprima esses dados na tela

2. Aprimore o programa anterior de forma que também seja imprimido na tela a média de idade e de peso desses alunos

Exercícios

1. Utilizando structures, projete um programa que receba nome, idade e o peso de 5 alunos e imprima esses dados na tela

2. Aprimore o programa anterior de forma que também seja imprimido na tela a média de idade e de peso desses alunos

3. Utilizando structures, projete um programa em que o usuário forneça os seguintes dados de n alunos:

- Data de Nascimento
- RA
- Ano de Ingresso na Universidade

e no final os imprima na tela.

PROJETINHO

Projetinho

- **Criar uma struct contendo as variáveis de nome, preço quantidade dos produtos que estarão disponíveis no sistema do mercado.**

```c
typedef struct {
    char* nome;
    float preco;
    int quantidade;
} Produto;

Produto produtos[5];
```

Projetinho

- Faça as alterações das variáveis para os respectivos campos da structure.
- Lembre de fazer a interação laço de repetição-structure para generalizar e deixar o programa mais otimizado.

MUITO OBRIGADO.

@petinfouem

pet@din.uem.br

petinformaticauem

discord.gg/5JaS4p4mWJ