Introdução a Programação

Revisão

Strings

main:

```
char str1[100];
```

main:

```
char str1[100];
char str2[3] = {‘P’, ‘E’, ‘T’};
```

main:

```
char str1[100];
char str2[3] = {‘P’, ‘E’, ‘T’};
char str3[] = “PET”;
```

main:

```
char str1[100];
char str2[3] = {‘P’, ‘E’, ‘T’};
char str3[] = “PET”;
char *str4;
```

funções

strcpy(s1, s2);
strcat(s1, s2);
strlen(s1);
strcmp(s1, s2);

funções

```
strcpy(s1, s2);     // copia s2 para s1
strcat(s1, s2);     // concatena s2 no final s1
strlen(s1);         // retorna o tamanho de s1
strcmp(s1, s2);     // compara s1 com s2
```

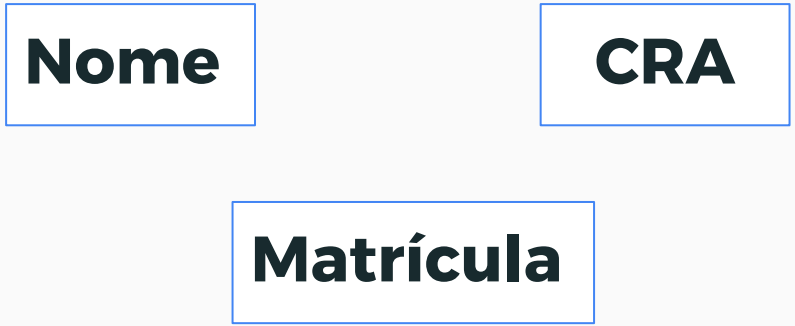
Structs & Unions

Structs

"Struct, ou estrutura, é um bloco que armazena diversas informações."

Variáveis

```
char *nome;
int matricula;
float CRA;
```



Estrutura

```
struct Aluno {
    char *nome;
    int matricula;
    float CRA;
};
```

main:

```
struct Aluno aluno;

aluno.nome = “Jane”;
aluno.matricula = 157;
aluno.CRA = 9.2;

printf(“%s %d %f”, aluno.nome, aluno.matricula,
aluno.CRA);
```

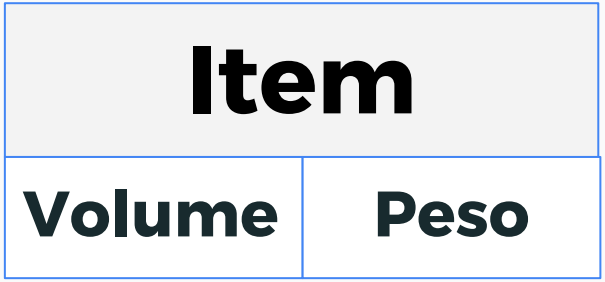
Usando *struct*, podemos trabalhar com vários tipos de informações de uma maneira mais fácil, rápida e organizada.

Unions

"Unions, ou união, é um bloco que armazena diversas informações no mesmo endereço."

Estrutura

```
struct Item {
    float volume;
    unsigned int peso;
};
```



União

```
union Item {
    float volume;
    unsigned int peso;
};
```

main:

```
union Item item;

item.volume = 2.3;
printf("%f", item.volume);

item.peso = 4;
printf("%f", item.peso);

printf("%p %p", &item.volume, &item.peso);
```

Union compartilha a memória para todos os elementos. A memória alocada é a do maior elemento contido nela.

Exercício