

## 3-UNION

- Já sabemos que quando fazemos uma declaração de uma variável, um espaço de memória é alocado para guardar o conteúdo da mesma.
- Cada tipo de dado ocupa um tamanho de espaço na memória, conseguimos verificar o quanto é esse espaço utilizando uma função chamada `sizeof()`.

```
#include <stdio.h>

int main() {
    printf("\n");

    int numero = 42;
    float nota = 7.9;
    char letra = 'd';

    printf("A variavel 'numero' tem valor %d e ocupa %ld bytes em memoria.\n", numero, sizeof(numero));
    printf("A variavel 'nota' tem valor %.2f e ocupa %ld bytes em memoria.\n", nota, sizeof(nota));
    printf("A variavel 'letra' tem valor %c e ocupa %ld bytes em memoria.\n", letra, sizeof(letra));

    printf("\n");
    return 0;
}
```

- Tanto faz você passar o nome da variável quanto seu tipo como parâmetro no `sizeof`. A variável ocupa o espaço em memória relativo ao dado.
- Outro detalhe é a colocação do `%ld` = long integer

```
A variavel 'numero' tem valor 42 e ocupa 4 bytes em memoria.

A variavel 'nota' tem valor 7.90 e ocupa 4 bytes em memoria.

A variavel 'letra' tem valor d e ocupa 1 bytes em memoria.
```

---

## UNION

Vamos criar um código.

```
#include <stdio.h>
#include <string.h> //acesso as funções de string

You, a few seconds ago | 1 author (You)
union pessoa{
    char nome[100]; // 1 char = 1 byte -> 1 * 100 = 100 bytes
    int idade; // 4 bytes
    // -> total de 104 bytes.
};

int main() {
    printf("\n");

    union pessoa pes; //union(nome[100], idade)

    strcpy(pes.nome, "Angelina Jolie"); //copiando a string para a variavel pes.nome
    printf("Dados de %s\n", pes.nome);

    pes.idade = 39;
    printf("Ela tem %d anos\n", pes.idade);

    printf("\nTAMANHO MEMORIA(pes) : %ld bytes", sizeof(pes));

    printf("\n");
    printf("\n");
    return 0;
}
```

- A union possui a mesma estrutura da struct, podemos criar a variavel tanto ao criar a union quanto dentro do main.

- **strcpy** é uma função que copia uma string para a variavel desejada.

- Depois veremos quanto a variavel possui de tamanho, pelos calculos deveria vir 4 bytes.

```
Dados de Angelina Jolie
Ela tem 39 anos

TAMANHO MEMORIA(pes) : 100 bytes
```

- Quando declaramos uma union, não importante quantas variaveis colocamos ali dentro, ela sempre irá ocupar somente o espaço da maior dessas variaveis.

```
You, a few seconds ago | 1 author (You)
union pessoa{
    char nome[100]; // 1 char = 1 byte -> 1 * 100 = 100 bytes
    int idade; // 4 bytes
    // -> total de 104 bytes.
};
```

- Assim, ela utiliza o espaço para outras variaveis, sobrescrevendo o anterior...

```
int main(){
    printf("\n");

    union pessoa pes; // union(nome[100], idade)

    strcpy(pes.nome, "Angelina Jolie"); // copiando a string para a variavel pes.nome
    printf("Dados de %s\n", pes.nome);

    pes.idade = 39;
    printf("Ela tem %d anos\n", pes.idade);

    printf("Dados de %s\n", pes.nome);

    printf("\nTAMANHO MEMORIA(pes) : %ld bytes", sizeof(pes));

    printf("\n");
    printf("\n");
    return 0;
}
```

```
Dados de Angelina Jolie
Ela tem 39 anos
Dados de '

TAMANHO MEMORIA(pes) : 100 bytes
```

- Ao tentar imprimir novamente o nome que foi adicionado, a memoria alocada não possui mais esse conteudo, logo não imprime...

- [0]

- [angelina jolie]

- [39], no final, so tem o ultimo valor.

---

OUTRO EXEMPLO

```

PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDENY\REP_UDENY\
PROG_C\512(struct_em C)\3-UNION> cmd /c .\p36.exe
0 valor de num1 eh : 9
0 valor de num2 eh : 9
0 valor de num3 eh : 9
0 valor de num4 eh : 9
0 valor de num5 eh : 9
PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDENY\REP_UDENY\
PROG_C\512(struct_em C)\3-UNION>

```

```

3 union numeros{
4     int num1, num2, num3, num4, num5;
5 }n;
6
7
8
9 int main(){
10
11     n.num1 = 1;
12     n.num2 = 3;
13     n.num3 = 5;
14     n.num4 = 7;
15     n.num5 = 9;
16
17     printf("\n0 valor de num1 eh : %d\n",n.num1);
18     printf("\n0 valor de num2 eh : %d\n",n.num2);
19     printf("\n0 valor de num3 eh : %d\n",n.num3);
20     printf("\n0 valor de num4 eh : %d\n",n.num4);
21     printf("\n0 valor de num5 eh : %d\n",n.num5);
22
23
24
25
26     return 0;
27 }

```

```

PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDENY\REP_UDENY\
PROG_C\512(struct_em C)\3-UNION> cmd /c .\p36.exe
0 valor de num1 eh : 1
0 valor de num2 eh : 3
0 valor de num3 eh : 5
0 valor de num4 eh : 7
0 valor de num5 eh : 9
0 valor de num1 eh : 9
0 valor de num2 eh : 9
0 valor de num3 eh : 9
0 valor de num4 eh : 9
0 valor de num5 eh : 9
'n' est[1] ocupando 4 bytes de memoria.
PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDENY\REP_UDENY\
PROG_C\512(struct_em C)\3-UNION>

```

```

9 union numeros{
10     int num1, num2, num3, num4, num5;
11 }n;
12
13 int main(){
14
15     n.num1 = 1;
16     printf("\n0 valor de num1 eh : %d\n",n.num1);
17     n.num2 = 3;
18     printf("\n0 valor de num2 eh : %d\n",n.num2);
19     n.num3 = 5;
20     printf("\n0 valor de num3 eh : %d\n",n.num3);
21     n.num4 = 7;
22     printf("\n0 valor de num4 eh : %d\n",n.num4);
23     n.num5 = 9;
24     printf("\n0 valor de num5 eh : %d\n",n.num5);
25     printf("\n0 valor de num1 eh : %d\n",n.num1);
26     printf("\n0 valor de num2 eh : %d\n",n.num2);
27     printf("\n0 valor de num3 eh : %d\n",n.num3);
28     printf("\n0 valor de num4 eh : %d\n",n.num4);
29     printf("\n0 valor de num5 eh : %d\n",n.num5);
30
31     printf("\n'n' está ocupando %ld bytes de memoria.\n",sizeof(n));
32
33     return 0;
34 }

```

## - Utilização:

- Temos 5 valores e vamos supor que precisamos alocar esses espaços de memoria para poder trabalhar. Teriamos a opção de declarar 5 variaveis, cada uma valendo 4 bytes dando um total de 20 bytes.

- Vamos supor que voce precise fazer uma soma de todos os numeros..Vamos ver quanto no total de memoria estamos ocupando:

```

PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDENY\REP_UDENY\
PROG_C\512(struct_em C)\3-UNION> cmd /c .\p36.exe
0 valor de num1 eh : 1
0 valor de num2 eh : 3
0 valor de num3 eh : 5
0 valor de num4 eh : 7
0 valor de num5 eh : 9
0 valor de num1 eh : 9
0 valor de num2 eh : 9
0 valor de num3 eh : 9
0 valor de num4 eh : 9
0 valor de num5 eh : 9
'n' esta ocupando 4 bytes de memoria.
soma' esta ocupando 4 bytes de memoria.
Soma = 25
Memoria total ocupada 8
PS C:\Users\Gabri\Documents\TEORIA_INDIVIDUAL\UDENY\REP_UDENY\
PROG_C\512(struct_em C)\3-UNION>

```

```

10
11 int soma = 0;
12
13 n.num1 = 1;
14 soma = soma + n.num1;
15 printf("\n0 valor de num1 eh : %d\n",n.num1);
16
17 n.num2 = 3;
18 soma = soma + n.num2;
19 printf("\n0 valor de num2 eh : %d\n",n.num2);
20
21 n.num3 = 5;
22 soma = soma + n.num3;
23 printf("\n0 valor de num3 eh : %d\n",n.num3);
24
25 n.num4 = 7;
26 soma = soma + n.num4;
27 printf("\n0 valor de num4 eh : %d\n",n.num4);
28
29 n.num5 = 9;
30 soma = soma + n.num5;
31 printf("\n0 valor de num5 eh : %d\n",n.num5);
32
33 printf("\n0 valor de num1 eh : %d\n",n.num1);
34 printf("\n0 valor de num2 eh : %d\n",n.num2);
35 printf("\n0 valor de num3 eh : %d\n",n.num3);
36 printf("\n0 valor de num4 eh : %d\n",n.num4);
37 printf("\n0 valor de num5 eh : %d\n",n.num5);
38
39 printf("\n'n' esta ocupando %ld bytes de memoria.\n",sizeof(n));
40 printf("\n'soma' esta ocupando %ld bytes de memoria.\n",sizeof(soma));
41
42 printf("\nsoma = %d\n",soma);
43
44 printf("\nMemoria total ocupada %ld\n",sizeof(n) + sizeof(soma));
45

```

- Estamos ocupando um total ded 8 bytes, logo vemos que compensa fazer uma union com varios numeros que precisamos pois estamos assim economizando de 8 para 24 bytes. 1/3 de memoria, do que fazendo a declaração no main, seja num array ou em duas variaveis.