FICHA 4: ESTRUTURAS LIGADAS E UNIÕES

MODALIDADE:	Educação e Formação de Adultos (EFA)	EFA NS (Profissional)	
CURSO:	Programador/a Informático/a		
UFCD:	Programação em C/C++ - avançado	CÓDIGO UFCD:	0810
FORMADOR/A:	Bruno Silva	DATA:	

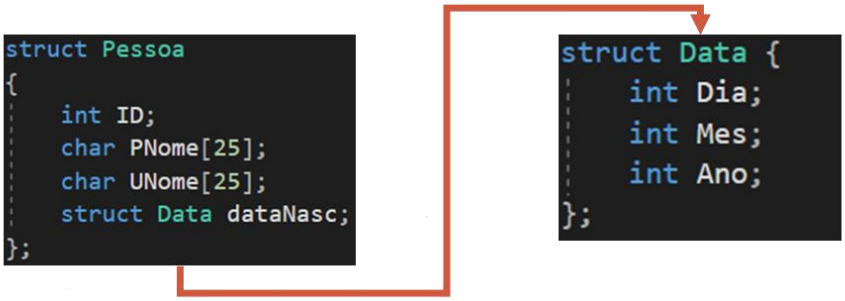
OBJETIVOS

- Saber como trabalhar com estruturas ligadas e uniões em C

Estruturas Ligadas

Como já referenciado anteriormente, uma estrutura pode conter, variáveis simples, vetores, ponteiros ou mesmo outras estruturas. Uma vez criada uma estrutura, pode-se definir outros tipos de estruturas e ligar uma ou mais estruturas, ou seja, estruturas ligadas (ou também podem ouvir a expressão estruturas encadeadas).

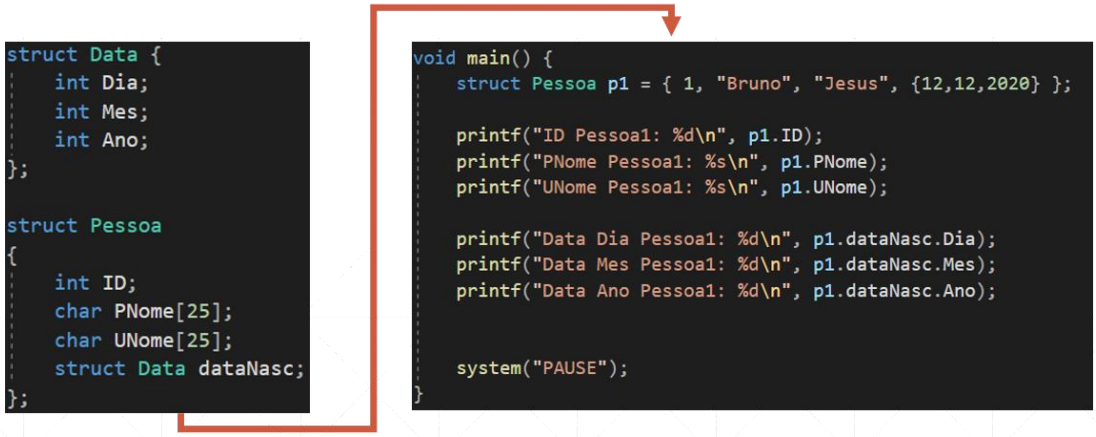
Exercício 1 – Vamos ***reaproveitar a estrutura que criamos anteriormente Pessoa*** e de seguida ***vamos incorporar uma estrutura dentro de outra***, mais especificamente o campo da ***Data*** na estrutura da ***Pessoa:***



```
struct Pessoa
{
    int ID;
    char PNome[25];
    char UNome[25];
    struct Data dataNasc;
};
```

```
struct Data {
    int Dia;
    int Mes;
    int Ano;
};
```

De seguida vamos fazer a declaração, carga inicial de valores e exibir os valores numa mensagem



```
struct Data {
    int Dia;
    int Mes;
    int Ano;
};

struct Pessoa
{
    int ID;
    char PNome[25];
    char UNome[25];
    struct Data dataNasc;
};
```

```
void main() {
    struct Pessoa p1 = { 1, "Bruno", "Jesus", {12,12,2020} };

    printf("ID Pessoa1: %d\n", p1.ID);
    printf("PNome Pessoa1: %s\n", p1.PNome);
    printf("UNome Pessoa1: %s\n", p1.UNome);

    printf("Data Dia Pessoa1: %d\n", p1.dataNasc.Dia);
    printf("Data Mes Pessoa1: %d\n", p1.dataNasc.Mes);
    printf("Data Ano Pessoa1: %d\n", p1.dataNasc.Ano);


    system("PAUSE");
}
```

Uniões

A palavra reservada ***union*** (união), serve para declarar estruturas especiais, tendo uma sintaxe de declaração muito parecida ao das estruturas.

A diferença entre as ***unions*** e as ***estruturas*** está na ***forma de armazenamento da informação***.

Embora que uma união possa armazenar tipos de dados diferentes (int, float, char, …)***, apenas será assumido 1 tipo de dados de cada vez***.

- Numa ***estrutura*** (***struct***), podemos armazenar uma variável do tipo int ***e*** outra variável do tipo char ***e*** outros tipos de dados, ***logo*** vários valores;
- Já na ***união*** (***union***) só pode armazenar uma variável do tipo int ***ou*** outra variável do tipo char ***ou*** outros tipos de dados, ***logo*** só armazena 1 valor;

Exercício 4 – Vamos ***reaproveitar a estrutura que criamos anteriormente Pessoa*** e de seguida ***modificar o último campo da estrutura pessoa***, mais especificamente o campo da ***Data*** na estrutura da ***Pessoa para o campo um_valor peso:***

```c
union um_valor {
    int val_int;
    float val_long;
    double val_double;
};

struct Pessoa
{
    int ID;
    char PNome[25];
    char UNome[25];
    union um_valor peso;
};
```

```c
void main() {
    struct Pessoa p1 = { 1, "Bruno", "Jesus" };

    printf("ID Pessoa1: %d\n", p1.ID);
    printf("PNome Pessoa1: %s\n", p1.PNome);
    printf("UNome Pessoa1: %s\n", p1.UNome);

    p1.peso.val_int = 15;
    p1.peso.val_double = 18.5;

    printf("Peso inteiro Pessoa1: %d\n", p1.peso.val_int);
    printf("Peso float Pessoa1: %f\n", p1.peso.val_long);
    printf("Peso double Pessoa1: %f\n", p1.peso.val_double);
```

Teste o programa anterior e veja os resultados na consola. Qual foi o valor que ficou associado a união? Foi o do valor valor_int 15 ou o valor val_double 18.5?