	INSTITUTO FEDERAL GOIANO – CÂMPUS URUTAÍ			
	STRUCT - REGISTRO			
	Curso:	GTI	Turma:	2º
	Disciplina:	LTP	Data:	01/02/21
	Professora:	**Vívian Cirino de Lima**		

Estrutura de Dados Composta Heterogênea - Estruturas, uniões e enumerados (Registros - Structs)

Observe o esquema abaixo:

1. **Estruturas de controle**

1.1. **Estrutura Sequencial –** os comandos são executados na sequência um após o outro, sem interrupção.

1.2. Estruturas Condicionais, de Decisão ou de Seleção – os comandos são executados de acordo com as situações definidas nas condições.

1.2.1. Estrutura Condicional Simples – uma condição (uso do *if*)
1.2.2. Estrutura Condicional Composta – existência de um bloco V (uso do *if)* e um bloco F (uso do *else*).
1.2.3. Estrutura Condicional Encadeada – existência de um agrupamento de condições, enquanto for necessário**.**
1.2.3.1. Estrutura de Seleção de Múltipla Escolha – Composta por uma série de estruturas de seleção simples encadeadas, em que observamos as seguintes prioridades:

- Todas as condições nas decisões são de igualdade.
- Todas as condições comparam uma mesma expressão a uma constante.
- Todas as constantes consideradas são de tipo inteiro ou caractere.

1.3. **Estruturas de Repetição –** Serve para repetir a execução de um bloco de comandos.
1.3.1. Repetição Contada – for
1.3.2. Repetição com precondição (while)
1.3.3. Repetição com poscondição (do/while)

2. **Estruturas de Dados Compostas**
2.1. Homogêneas – Vetor e Matriz (Arrays)
2.2. Heterogêneas – Registro (Struct)

Nas aulas anteriores trabalhamos com estruturas primitivas e de tipos agregados homogêneos. Contudo, a necessidade de trabalhar com vários tipos de dados em um conjunto é sempre sentida pelo programador. Quando usamos matrizes, notamos que somente foi possível trabalhar com um tipo de dado por matriz. Para solucionar essa deficiência, será utilizado uma struct, a qual permite

trabalhar com vários dados de tipos diferentes em uma mesma tabela. Por essa razão, esse dado é heterogêneo.

Imagine trabalhar com variáveis que contém dados de funcionários de uma empresa. Código do funcionário, nome, endereço, ano de admissão, data de nascimento; são somente alguns dos muitos dados que se poderia querer guardar. Imagine agora trabalhar com centenas de funcionários, seria muito complicado manipular todas estas variáveis soltas. Mesmo com a utilização de vetores ou matrizes ainda teríamos que definir diversas estruturas homogêneas para cada tipo de variável que se desejasse usar como um vetor para nomes, outro para endereços, outro para datas, etc.

É comum que uma linguagem de programação apresente métodos de agregação de variáveis heterogêneas. Isto significa na prática uma variável formada por várias outras variáveis de tipos diversas.

Hoje veremos como a linguagem C possibilita a implementação destes agregados heterogêneos. Veremos também como é simples e versátil combinar agregados homogêneos de dados (vetores e matrizes) com agregados heterogêneos (registros).

Sintaxe:

struct <nome do identificador> //nome dado a estrutura
{
	<lista dos tipos>
};
struct <nomedoidentificador> <variável>; //variável – utilizada para chamar os membros da estrutura.

Ex:
```
struct DADOS
{
	char nome[30];
	int id;
};
struct DADOS PESSOA;
```

Uma struct é uma variável especial que contém diversas outras variáveis normalmente de tipos diferentes.

As variáveis internas contidas pela struct são denominadas membros da struct.

Podemos dizer que as structs da linguagem C são o equivalente ao que se denomina registros em outras linguagens de programação.

Sintaxe:

```
struct <identificador>
{
 <listagem dos tipos e membros>;
}
struct <identificador> <variavel>;
```

Exemplo de declaração de uma struct

```
struct ficha_de_aluno
{
  char nome[50];
  char disciplina[30];
  float nota_prova1;
```

```
  float nota_prova2;
};
```

struct ficha_de_aluno aluno;

Neste exemplo criamos a struct ficha_de_aluno.

Depois de criar a struct precisamos criar a variável que vai utiliza-la.

Para tanto criamos a variável aluno, que será do tipo ficha_de_aluno.

struct ficha_de_aluno aluno;

Agora vejamos um programa que utiliza uma struct.

```
#include <stdio.h>
#include <conio.h>
int main(void)
{ /
*Criando a struct */
  struct ficha_de_aluno
  {
    char nome[50];
    char disciplina[30];
    float nota_prova1;
    float nota_prova2;
  };

   /*Criando a variável aluno que será do
     tipo struct ficha_de_aluno       */
  struct ficha_de_aluno aluno;

  printf("\n---------- Cadastro de aluno -----------\n\n\n");
   printf("Nome do aluno ......: ");
  __fpurge(stdin);

  /*usaremos o comando fgets() para ler strings, no caso o nome
   do aluno e a disciplina
   fgets(variavel, tamanho da string, entrada)
Como estamos lendo do teclado a entrada é stdin (entrada
padrão), porém em outro caso, a entrada tambem poderia ser um
arquivo  */

  fgets(aluno.nome, 40, stdin);
   printf("Disciplina ......: ");
  __fpurge(stdin);
  fgets(aluno.disciplina, 40, stdin);
   printf("Informe a 1a. nota ..: ");
  scanf("%f", &aluno.nota_prova1);
  printf("Informe a 2a. nota ..: ");
  scanf("%f", &aluno.nota_prova2);
  printf("\n\n --------- Lendo os dados da struct--------\n\n");
  printf("Nome ...........: %s", aluno.nome);
  printf("Disciplina .....: %s", aluno.disciplina);
  printf("Nota da Prova 1 ...: %.2f\n" , aluno.nota_prova1);
```

```
    printf("Nota da Prova 2 ...: %.2f\n", aluno.nota_prova2);
    return(0);
}
```

https://www.youtube.com/watch?v=k7L_5PAUxmk

https://www.youtube.com/watch?v=U5bfBqFymp4 (estudar os exemplos e projetar para os alunos no GEANY)