

Aula 09 - Estruturas

Assuntos

1. Enum
2. Struct
3. Union

Enumeração (Enum)

Server para enumerar uma sequência de valores

```
#include <stdio.h>

int main(void) {
    enum dias_semana {
        segunda, terça, quarta, quinta, sexta, sabado, domingo
    };

    printf("%d %d %d %d %d %d %d\n",
        segunda, terça, quarta, quinta, sexta, sabado, domingo);
}
```

Pelo `printf` podemos ver que um `enum` é só uma sequência de inteiros, começando do zero.

Enum

Se um `enum` é só uma sequência de números, então:

- para o que pode ser usado?
- qual a vantagem em usar no lugar de ter apenas o número?

Enum - documentação de erro

```
#include <stdio.h>

int main(void) {
    enum codigos {OK, ERRO_PRINT};
    enum codigos cod = OK;

    int len = printf("hello, world\n");
    if (len < 0)
        cod = ERRO_PRINT;

    return cod;
}
```

Enum

Também é possível forçar um valor para o início do enum

```
#include <stdio.h>

int main(void) {
    enum meses {
        janeiro=1, fevereiro, marco,
        abril, maio, junho,
        julho, agosto, setembro,
        outubro, novembro, dezembro};

    printf("%d %d\n", janeiro, dezembro);
}
```

Struct

Permitem agrupar um conjunto de variáveis dentro de um tipo (estrutura), sendo que cada variável pode ter um tipo diferente.

Struct

Podemos definir um tipo da estrutura e depois instanciar variáveis desse tipo.

```
struct nome_da_estrutura {  
    int i;  
    float f;  
    char c;  
};  
  
struct nome_da_estrutura nome_da_variavel;
```


Struct

Podemos definir a estrutura e já criar as variáveis que são desse tipo.

```
// define uma variável com a estrutura definida
struct {
    int i;
    float f;
    char c;
} nome_da_variavel;
```

Struct - exemplo

```
#include <stdio.h>
#include <string.h>

struct disciplina {
    int aulas_teorica;
    int aulas_lab;
    char codigo[7];
    char nome[50];
};
```

Struct - exemplo

```
int main(){  
    struct disciplina cc2632;  
    strcpy(cc2632.codigo, "CC2632");  
    strcpy(cc2632.nome, "Desenvolvimento de Algoritmos");  
    cc2632.aulas_teorica = 2;  
    cc2632.aulas_lab = 2;  
}
```

Struct - exemplo

```
printf("\nValores:\n");  
printf("Codigo: %s\n", cc2632.codigo);  
printf("Nome: %s\n", cc2632.nome);  
printf("Aulas de teoria: %d\n", cc2632.aulas_teorias);  
printf("Aulas de laboratorio: %d\n", cc2632.aulas_lab);  
}
```

Struct - exemplo de leitura do teclado

```
struct disciplina d;  
  
char codigo[7];  
scanf("%s", codigo);  
strcpy(d.codigo, codigo);  
  
char nome[50];  
scanf("%s", &d.nome);  
strcpy(d.nome, nome);  
  
scanf("%d", &d.aulas_teorica);  
scanf("%d", &d.aulas_lab);
```

Struct - exemplo de leitura do teclado

```
printf("\nValores:\n");  
printf("Codigo: %s\n", d.codigo);  
printf("Nome: %s\n", d.nome);  
printf("Aulas de teoria: %d\n", d.aulas_teorias);  
printf("Aulas de laboratorio: %d\n", d.aulas_lab);
```

Struct - exemplo de passagem para uma função

Vamos ver com calma nas próximas aulas porque depende de ponteiros.

Struct - exemplo de passagem para uma função

```
#include <stdio.h>

struct disciplina {
    int aulas_teorica;
    int aulas_lab;
    char *codigo;
    char *nome;
};
```


Struct - exemplo de passagem para uma função

```
void le_valores(struct disciplina *d){  
    char codigo[7];  
    scanf("%s", codigo);  
    strcpy(d->codigo, codigo);  
  
    char nome[50];  
    scanf("%s", nome);  
    strcpy(d->nome, nome);  
  
    scanf("%d", &d->aulas_teorias);  
    scanf("%d", &d->aulas_lab);  
}
```

Struct - exemplo de passagem para uma função

```
int main(void) {  
  
    struct disciplina d;  
  
    le_valores(&d);  
  
    printf("\nValores:\n");  
    printf("Codigo: %s\n", d.codigo);  
    printf("Nome: %s\n", d.nome);  
    printf("Aulas de teoria: %d\n", d.aulas_teorica);  
    printf("Aulas de laboratorio: %d\n", d.aulas_lab);  
}
```

Struct + vetores

Se o struct se torna um tipo que podemos instanciar, então podemos ter um array de structs

```
struct disciplina d[10];  
  
le_valores(&d[0]);  
  
printf("Codigo: %s\n", d[0].codigo);  
printf("Nome: %s\n", d[0].nome);  
printf("Aulas de teoria: %d\n", d[0].aulas_teorias);  
printf("Aulas de laboratorio: %d\n", d[0].aulas_lab);
```

Struct + typedef

Em C podemos usar o `typedef` para facilitar um pouco o uso de structs:

```
struct StructPonto{  
    int x;  
    int y;  
};  
  
typedef struct StructPonto Ponto;  
  
Ponto p;
```

Struct + typedef

```
typedef struct {  
    int x;  
    int y;  
} Ponto;
```

```
Ponto p;
```

Union

Funcionam como um `struct`, porem as variáveis ocupam o mesmo espaço na memória

```
#include <stdio.h>

int main(void) {
    union teste {
        int i;
        float f;
    } t;

    t.i = 1;
    t.f = 2;

    printf("%d %f", t.i, t.f);
}
```