

UNIVERSIDADE ESTADUAL DE MS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
AEDI

Struct / Enum / Union

Prof. Nilton

Struct

- *Struct* é um agregado de variáveis relacionadas entre si sob um mesmo nome.
- *Struct* pode conter variáveis de tipos diferentes;
- Sintaxe geral:

```
struct  identificador {  
    tipo  nome_variável;  
    ...  
    tipo  nome_variável;  
} variável_tipo_estrutura;
```

Declarando o registro DATA

```
struct {  
    int dia;  
    int mes;  
    int ano;  
} DATA; //var DATA
```

Usando o registro DATA

```
...  
scanf("%d",&DATA.dia);  
printf("\n%d",DATA.dia);  
...
```

Outro exemplo de registro

```
struct {  
    int dia;  
    int mes;  
    char aniversariantes[100][50];  
} NIVER[366];
```

Usando o registro NIVER

```
...  
NIVER[99].dia=9;  
NIVER[99].mes=4;  
strcpy(NIVER[99].aniversariantes[0],"FULANO");  
...
```

Definindo uma classe de registros para um mesmo tipo

```
struct TDATA {    //identificador TDATA  
    int dia;  
    int mes;  
    int ano;  
};
```

...

```
struct TDATA DATA; //var DATA
```

```
scanf("%d",&DATA.dia);  
printf("\n%d",DATA.dia);
```

...

Classe de registros como parâmetros

```
struct TDATA {  
    int dia;  
    int mes;  
    int ano;  
};  
  
void le(struct TDATA *d) {  
    scanf("%d",&d->dia);  
}  
  
void escreve(struct TDATA d) {  
    printf("\n%d",d.dia);  
}  
  
int main(void) {  
    struct TDATA DATA;  
  
    le(&DATA);  
    escreve(DATA);  
    ...  
}
```

Criando um novo tipo e passagem de parâmetros

```
struct RDATA {  
    int dia;  
    int mes;  
    int ano;  
};
```

```
typedef struct RDATA TIPO_DATA;
```

```
void le(TIPO_DATA *d) {  
    scanf("%d",&d->dia);  
}
```

```
void escreve(TIPO_DATA d) {  
    printf("\n%d",d.dia);  
}
```

```
int main(void) {  
    TIPO_DATA DATA;  
  
    le(&DATA);  
    escreve(DATA);  
    ...  
}
```

Criando um novo tipo de outra maneira

```
typedef struct RDATA {  
    int dia;  
    int mes;  
    int ano;  
} TIPO_DATA;
```

```
void le(TIPO_DATA *d) {  
    scanf("%d",&d->dia);  
}
```

```
void escreve(TIPO_DATA d) {  
    printf("\n%d",d.dia);  
}
```

```
int main(void) {  
    TIPO_DATA DATA;
```

```
    le(&DATA);  
    escreve(DATA);
```

```
    ...
```


Conjunto de registros a partir de um novo tipo

```
typedef struct RDATA {  
    int dia;  
    int mes;  
    int ano;  
} TIPO_DATA;
```

```
void le(TIPO_DATA d[100]) {                // ou   d[]   ou   *d  
    scanf("%d",&d[5].dia);  
}
```

```
void escreve(TIPO_DATA d[100]) {           //ou   d[]   ou   *d  
    printf("\n%d",d[5].dia);  
}
```

```
int main(void) {  
    TIPO_DATA DATA[100];  
  
    le(DATA);  
    escreve(DATA);  
    ...  
}
```

Tipo de conjunto

```
typedef struct RDATA {  
    int dia;  
    int mes;  
    int ano;  
} TIPO_DATA[100];
```

```
void le(TIPO_DATA d) {  
    scanf("%d",&d[5].dia);  
}
```

```
void escreve(TIPO_DATA d) {  
    printf("\n%d",d[5].dia);  
}
```

```
int main(void) {  
    TIPO_DATA DATA;  
  
    le(DATA);  
    escreve(DATA);  
    ...  
}
```

Mais sobre *typedef*

```
typedef int inteiro;  
typedef enum{FALSE, TRUE} logico;  
typedef signed short int sshort;
```

```
int main(void) {  
    inteiro a=0, b=0;  
    logico passou=FALSE;  
    sshort my=32767;
```

```
    printf( "\n%d %d %d %d", a, b, passou, my );
```

0 0 0 32767

```
    a = a + 1;
```

```
    b = b + 1;
```

```
    my = my + 1;
```

```
    passou = TRUE;
```

```
    printf( "\n%d %d %d %d", a, b, passou, my );
```

1 1 1 -32768

```
    ...
```

Enumerações

```
enum meses {JAN = 1, FEV, MAR, ABR, MAI, JUN, JUL, AGO, SET,  
            OUT, NOV, DEZ};
```

Este comando faz duas coisas:

1. Torna meses o nome de um novo tipo; meses é chamado de enumeração;
2. Estabelece JAN, FEV, MARC, etc. como constantes simbólicas para os inteiros 1-12. Estas constantes são chamadas de enumeradores;

OBS

Se o primeiro enumerador não for inicializado o valor default é 0 e os demais +1;

Enumeradores podem qualquer valor de inicialização

```
enum bits {um = 1, dois = 2, quatro = 4, oito = 8};
```

Exemplo de enum sem definição de tipo

```
enum meses {JAN = 1, FEV, MAR, ABR, MAI, JUN, JUL, AGO, SET,  
            OUT, NOV, DEZ};
```

...

```
enum meses mes;
```

```
char nomeMes[][15] = {"", "Janeiro", "Fevereiro", "Marco", "Abril",  
                      "Maio", "Junho", "Julho", "Agosto", "Setembro", "Outubro",  
                      "Novembro", "Dezembro"};
```

```
for (mes = JAN; mes <= DEZ; mes++)  
    printf("%2d %s\n", mes, nomeMes[mes]);
```

...

```
1 Janeiro  
2 Fevereiro  
3 Marco  
4 Abril  
5 Maio  
6 Junho  
7 Julho  
8 Agosto  
9 Setembro  
10 Outubro  
11 Novembro  
12 Dezembro
```

Union

Union é um formato de dados que pode armazenar tipos diferentes, mas um tipo de cada vez;

```
union umpratodos {  
    int val_int;  
    double val_double;  
};
```

```
union umpratodos c;
```

```
c.val_int = 15;      //armazena um int  
printf("%d\n\n", c.val_int);  
c.val_double = 1.38; //armazena um double, e o int perde-se  
printf("%g\n\n", c.val_double);
```

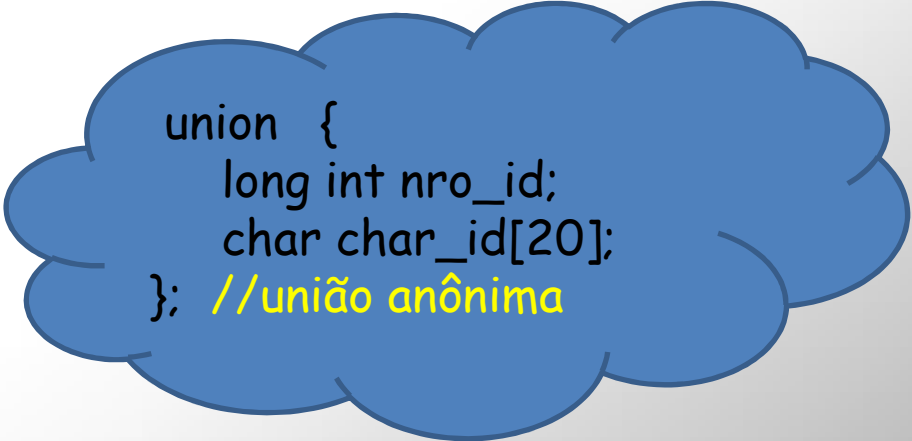
Struct e Union

```
struct inventario {  
    char marca[20];  
    int tipo;  
    union {                // formato depende do tipo inventario  
        long int nro_id;    // inventários do tipo 1  
        char char_id[20];  // outros inventários  
    } id;  
};
```

...

```
struct inventario preco;
```

```
printf("Qual o tipo? (1 para inteiro): ");  
scanf("%d", &preco.tipo);  
if (preco.tipo == 1){  
    printf("Entre com o numero: ");  
    scanf("%ld", &preco.id.nro_id);  
} else {  
    printf("Entre com o nome: ");  
    scanf("%s", preco.id.char_id);  
}
```



```
union {  
    long int nro_id;  
    char char_id[20];  
}; //união anônima
```

Usando union para transformar um character ASCII em binário

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

```
#include <conio.h>
```

```
struct byte {
```

```
    int a : 1;
```

```
    int b : 1;
```

```
    int c : 1;
```

```
    int d : 1;
```

```
    int e : 1;
```

```
    int f : 1;
```

```
    int g : 1;
```

```
    int h : 1;
```

```
};
```

```
union bits {
```

```
    char ch;
```

```
    struct byte bit;
```

```
} ascii;
```


Usando union para transformar um character ASCII em binário

```
void printa(int i) {  
    if(i)  
        printf("1 ");  
    else  
        printf("0 ");  
}
```

```
void decode(union bits b) {  
    printa(b.bit.h);  
    printa(b.bit.g);  
    printa(b.bit.f);  
    printa(b.bit.e);  
    printa(b.bit.d);  
    printa(b.bit.c);  
    printa(b.bit.b);  
    printa(b.bit.a);  
    printf("\n");  
}
```

Usando union para transformar um character ASCII em binário

```
int main() {  
  
    do {  
        ascii.ch = getche();  
        printf(": ");  
        decode(ascii);  
    } while(ascii.ch != 'q');  
    return 0;  
}
```

Exercícios

1- Defina um registro capaz de armazenar as seguintes informações sobre um determinado cliente de um banco: nome, CPF, RG, número da conta, data de abertura da conta e saldo. A data deve ser definida também como um registro composto de dia, mês e ano.

2- Dados os seguintes campos de um registro: nome, telefone, dia de aniversário e mês de aniversário, desenvolver um programa que mostre em um dado mês do ano, quem são as pessoas que fazem aniversário, exibir também o dia. Considere um conjunto de 40 pessoas.

3- Uma pessoa cadastrou um conjunto de 15 registros contendo o nome da loja, telefone e preço de um eletrodoméstico. Desenvolver um programa que permita exibir qual foi a média dos preços cadastrados e uma relação contendo o nome e o telefone das lojas cujo preço estava abaixo da média.

Exercícios

4- Suponha um cadastro de participantes com um registro contendo Nome e CPF do aluno, tipo de participação (A, B, C ou D) e sócio da SBC (S-sim ou N-não). Desenvolver um programa para calcular o valor que cada aluno vai pagar para participar do **SBBD 2019**, sabendo-se que:

Tipo de Participação	Valor a Pagar
A - 1 curso	R\$ 30,00
B - 2 cursos	R\$ 60,00
C - 3 cursos	R\$ 90,00
D - outros	R\$100,00

Para os sócios da SBC o valor a pagar terá um desconto de 50%. O programa deverá permitir a entrada de vários registros (no máximo 1000) até que uma condição de finalização seja satisfeita. Calcular e exibir também o total geral arrecadado com o evento e quantos alunos se matricularam em cada um dos tipos de participação.

Exercícios

5- Produtos são comercializados por uma loja que precisa manter as informações como nome do produto, código do produto, nomes de fornecedores, quantidade mínima em estoque, quantidade real em estoque, ultima data de venda e de compra e qual fornecedor fez o último fornecimento. A loja mantém 50 produtos diferentes e cada produto pode ser fornecido por até 4 fornecedores. Construa um programa que:

- a) Tenha módulos para inicialização do conjunto de registros e para cadastro dos produtos;
- b) Tenha um módulo para imprimir quais produtos cuja quantidade real em estoque é inferior ou igual a quantidade mínima em estoque. Deve ser impresso o nome do produto, seus fornecedores, o fornecedor que fez último fornecimento e as quantidades em estoque;
- c) Tenha um módulo que determine quantos produtos um determinado fornecedor fornece. Um dos parâmetros do módulo é o fornecedor;
- d) Determine os códigos dos produtos fornecidos por cada fornecedor. Use tantos módulos que forem necessários;