# Engenharia de Computação

# Fundamentos de Computação

## Aula 9 – Tipos definidos pelo Usuário: Struct , Union, Enum, Typedef

**Prof. Fernando Barreto**

informatica-ap@utfpr.edu.br

# Struct



- Um **struct** é um esquema/padrão que define um grupo de várias variáveis com alguma relação em comum
  - Exemplo: Cadastro de clientes que possui, nome, endereço e idade.

Cadastro do cliente X
- Nome: Fulano da Silva
- Endereço: Rua sei lá, 555.
- Idade: 30

Cadastro do cliente Y
- Nome: Zé da Árvore
- Endereço: Casa da Mata, 13
- Idade: 76

  - As definições de Nome, Endereço, Idade acima têm uma relação, no caso para representar informações de um cadastro de cliente (cliente X e cliente Y)
  - Nesse caso utilizamos um **struct**

# Definição - Struct



```
struct esquema_cadastro {
      char nome[21];
      char endereco[31];
      int idade;
};
```

- **Definição do molde/esquema** da struct, deve ser após os includes!!
- A definição de uma struct por si só **não** é uma declaração de variável !
- Não é possível inicializar
- Observar ; no final do }

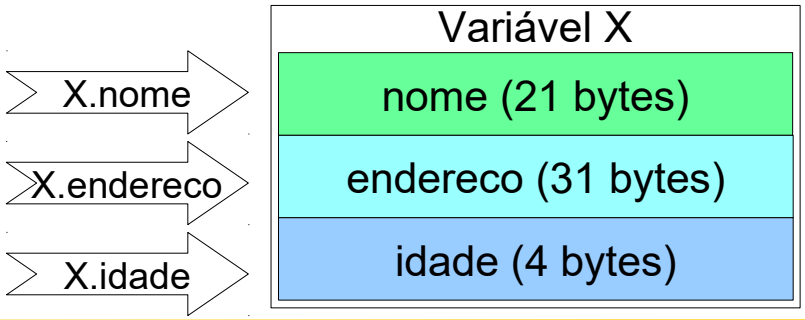
```
struct esquema_cadastro X;
```

- **Declaração** da variável **X** que será tipo **struct esquema_cadastro**

```
int main( )
{
   fgets(X.nome,sizeof(X.nome),stdin);
   fgets(X.endereco,sizeof(X.endereco),stdin);
   scanf("%i",&X.idade);

   printf("Nome: %s , Endereço: %s , Idade: %i", X.nome, X.endereco, X.idade);
   return 0;
}
```

A referência para uma variável interna de X na estrutura utiliza "."
- Exemplo: **X.idade**



# Struct



- Pode-se trabalhar com vetores ou matrizes de struct
- Exemplo utilizando a estrutura anterior para declarar um vetor de 50 cadastros de clientes do tipo struct esquema_cadastro:

```
struct esquema_cadastro cadastros[50];
```

  - Acesso a uma variável interna da struct dentro de um vetor/matriz necessita do "." :

```
cadastros[indice].variavel_interna
```

  - Exemplos:

```
if (strcmp(cadastros[13].nome, cadastro[10].nome)==0) {
    printf("Cadastros com nomes iguais!");
}
strcpy(cadastros[0].endereco,cadastros[30].endereco);
cadastros[13].idade = 40;
```

# Struct



- Exemplo de inicialização de uma variável vetor de 3 structs esquema_cadastro

```
struct esquema_cadastro {
        char nome[21];
        char endereco[31];
        int idade;
};

struct esquema_cadastro cadastros[3] = {
       { "Paulo Teste", "Rua do Torto, 1234", 56 }, //índice 0 do vetor
       { "Ze Roberto", "Rua dos Robertos, 4321", 65 }, //índice 1 do vetor
       { "Maria Carla", "Av. Sem Fundo, 778", 28 }   //índice 2 do vetor
};
```

# Struct



- Operações com variáveis struct
  - Permite atribuição direta entre variáveis da mesma struct

```
struct esquema_cadastro {
        char nome[21];
        char endereco[31];
        int idade;
};

struct esquema_cadastro cadastros[3] = {
       { "Paulo Teste", "Rua do Torto, 1234", 56 }, //índice 0 do vetor
       { "Ze Roberto", "Rua dos Robertos, 4321", 65 }, //índice 1 do vetor
       { "Maria Carla", "Av. Sem Fundo, 778", 28 }   //índice 2 do vetor
};

int main( ) {
      cadastros[0] = cadastros[1];  //irá copiar todos os itens internos (nome, endereco, idade)
      return 0;
}
```

# Struct



- Structs aninhados
  - Permite utilizar uma definição de struct dentro de outra

```
struct esquema_endereco {
        char nomeRua[31];
        int numero;
};

struct esquema_cadastro {
        char nome[21];
        struct esquema_endereco end;
        int idade;
};

int main( ) {
      struct esquema_cadastro X;
      X.end.numero=1234; //exemplo de como acessar variavel struct aninhado

      return 0;
}
```

# Exercícios Struct



1) Crie uma estrutura para representar as coordenadas de um ponto no plano (posições X e Y). Em seguida, declare e leia do teclado dois pontos e exiba a distância entre eles.

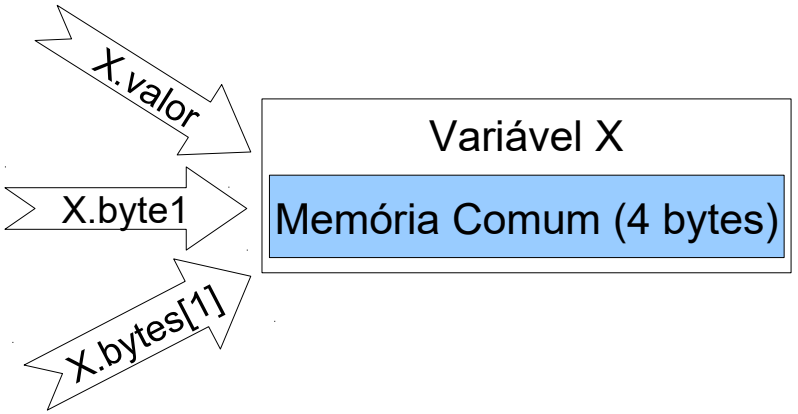
2) Crie um programa utilizando struct para receber e armazenar 4 notas de 3 alunos identificados por nome, idade e media. A média aritmética é calculada depois de preencher os cadastros. Em seguida, mostre uma listagem do nome dos alunos, idade, média e, depois, qual o aluno tem a maior média.

# Union



- Aloca uma região de memória que é interpretada de acordo com o acesso feito pelos tipos internos definidos na union
  - Região de memória compartilhada entre tipos de variáveis diferentes…
  - Ao modificar uma variável da Union, afeta-se as demais...
- Aloca-se memória para acomodar o maior tipo definido
- Tanto a declaração quanto acesso são similares à struct
- Usos comuns: área de memória que podem assumir várias funções (ex: registradores de TX ou RX de placa de rede)

```
union exemplo {
        int valor; //tamanho de 4 bytes
        char byte1; //tamanho de 1 byte
        char bytes[4]; //tamanho de 4 bytes
};
int main(){
        union exemplo X;
        X.valor = 1024; //modificará 4 bytes
        X.byte1 = 'a'; //modifica 1 byte e X.valor terá outra informação agora!!!!
}
```



# Enum



- Permite definir uma lista de constantes inteiros
  - Usar maiúsculas (boa prática)
  - Se não inicializado, a 1ª constante = 0 e as demais = 1,2,3,4…..
  - Duas ou mais constantes podem ter o mesmo valor

Lista de constantes

```
enum semana { DOM, SEG, TER, QUA, QUI, SEX, SAB };
int main(){
      enum semana X; //X pode ser um item da lista de constantes de semana
      X = DOM; //X recebe valor 0
      X = QUI; //X recebe valor 4
}
```

```
enum teste { A=6, B, C=10, D, E=10, F, G};
int main(){
      enum teste X;
      X = B; //X recebe valor 7
      X = G; //X recebe valor 12
}
```

# Typedef



- Definir seus próprios tipos com base em outros previamente definidos
  - Seria um "sinônimo" para um tipo existente
  - Exemplo de uso: diferenciar tamanhos de inteiros e outros tipos devido à arquitetura
    - *https://github.com/esp8266/Arduino/blob/master/tools/sdk/include/c_types.h*
  - Outro exemplo prático: structs

```
struct meuCadastro {
        char nome[21];
        char endereco[31];
        int idade;
} ;

int main( ) {
      struct meuCadastro X[10];
      return 0;
}
```

```
typedef struct {
        char nome[21];
        char endereco[31];
        int idade;
} meuCadastro;

int main( ) {
      meuCadastro X[10];
      return 0;
}
```

O nome da struct **pode** ser omitido ao usar typedef

Tipo definido: meuCadastro

# Exercício



3) Crie um programa de cadastro para armazenar até 10 pessoas. O cadastro (utilizar struct e typedef) deve conter o nome, idade, estado civil (utilizar enum), endereço IP.

- Deve-se planejar logicamente um cadastro vazio como a idade contendo -1.
- O endereço IP poderá ser lido como um inteiro não sinalizado ou como 4 bytes em char não sinalizado (usar union).
- O programa deve conter um menu com opções de (1) inserir pelo índice (o usuário escolhe qual posição do vetor ele quer inserir no cadastro), (2) remover pelo índice, (3) listar todos os cadastrados não vazios e (4) sair.