

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Estruturas

Grupos de variáveis relacionadas entre si (podendo ser de tipos diferentes) sob um mesmo nome.

Exemplo: `struct carta {
 char*face;
 char*naipe;
};`

É criada uma estrutura do tipo **struct carta** que contém duas variáveis do tipo **char**.

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Exemplos:

```
struct a {  
    float num1;  
    int num2;  
};  
  
struct agenda {  
    char*nome;  
    char*sobrenome;  
    char*endereço;  
    int telefone;  
    float altura;  
};  
  
struct carta {  
    char*face;  
    char*naipe;  
};
```

| | |
|--------------------|--|
| struct | definição |
| carta | identificador (rótulo) |
| face, naipe | membros (tipos básicos, arrays, estrutura) |

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

```
struct agenda {  
    char*nome;  
    char*sobrenome;  
    char*endereço;  
    int telefone;  
    float altura;  
};
```

A própria estrutura não pode ser um membro, mas pode-se ter um ponteiro para ela → Estrutura auto-referenciada

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Declaração:

```
int a, baralho [52], *cptr;
```

Declaração do tipo **int**

ou

Declaração do tipo **struct carta**

```
struct carta {  
    char*face;  
    char*naipe;  
} a, baralho [52], *cptr;
```

Ponteiro para uma estrutura do tipo **carta**

Matriz com 52 elementos do tipo **carta**

Estrutura do tipo **carta** com o nome **a**

O identificador é opcional, porém a declaração de outras variáveis do tipo **carta** só será possível com a definição do identificador

```
struct carta {  
    char*face;  
    char*naipe;  
};
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Operações:

- Atribuição entre variáveis de um mesmo tipo de estrutura;
- Obtenção do endereço de um membro;
- Acesso aos membros;
- Determinação do tamanho (`sizeof`)

Inicialização :

```
struct carta a = {"tres", "copas"};
```

Se houver menos inicializadores que membros estão ao restante será atribuído zero ou NULL(ponteiro).

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Acesso a membros da estrutura:

- Operador ponto(.)
- Operador de ponteiro de estrutura (->)

- Ex:

```
printf ("%s", a.naipes);  
printf ("%s", cptr -> naipes);  
printf ("%s", (*cptr). naipes);
```

cptr é um ponteiro e aponta para **struct carta**

```
struct carta {  
    char*face;  
    char*naipes;  
} a, baralho [52], *cptr;
```

Usando Estruturas com Funções:

- Passagem de uma estrutura: passagem por valor
- Passagem de um membro: passagem por valor
- Passagem de um ponteiro para estrutura: passagem por referência

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Typedef:

“Criação virtual” de novos tipos → sinônimos (alias)

Ex: `typedef struct carta CARTA;` cria o tipo **CARTA**

ou `typedef struct {
 char *face;
 char *nape;
} CARTA;`

Cria o tipo **Carta** (uma estrutura)

Utilização: `CARTA baralho[52];`

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Exemplo de uso:

```
#include<stdio.h>

struct carta {
    char *face;
    char *naipe;};

main ( )
{
    struct carta A;
    struct carta *Aptr;
    A.face = "As";
    A.naipe = "espadas";
    Aptr = &A;
    printf ("%s\n%s\n%s\n\n",A.face,
            Aptr->face, (*Aptr).face);

    system("pause");
}
```

As
As
As

Pressione qualquer tecla :

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

UNIÕES

É um tipo derivado de dados (como uma estrutura) cujos membros compartilham o mesmo espaço de armazenamento.

Também deve ser suficiente para acomodar o maior membro.
Apenas um membro pode ser acessado por vez.

Declaração

```
union numero {  
    int x;  
    float y;  
};
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Idéia do uso de memória:

Estrutura

União

float

int

float
ou
int

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Operações permitidas

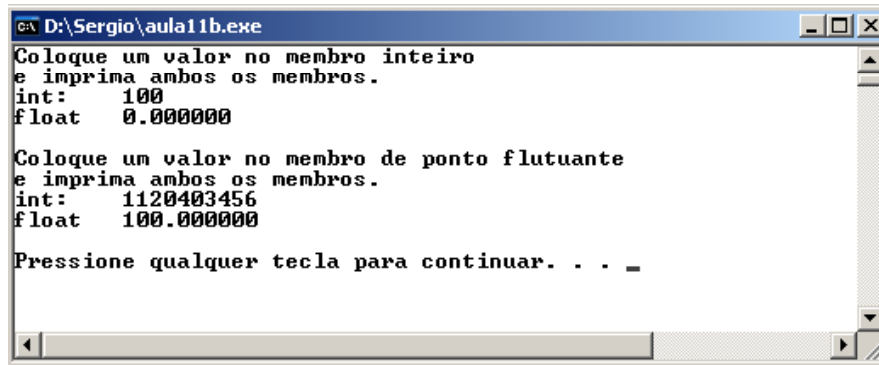
- Atribuição (atribuir uma união a outra união)
- Obtenção do endereço
- Acesso aos membros

Inicialização: Apenas com um valor do mesmo tipo que o do primeiro membro da união.

```
union numero valor={10};    union numero {  
                                int x;  
                                float y;  
                                };  
  
Essa declaração seria inválida;  
union numero valor={1.43};
```

```
union numero{  
    int x;  
    float y;  
};  
  
main()  
{ system("color f0");  
  union numero valor;  
  
  valor.x = 100;  
  printf("%s\n%s\n%s%d\n%s%f\n\n",  
         "Coloque um valor no membro inteiro",  
         "e imprima ambos os membros.",  
         "int:   ", valor.x,  
         "float  ", valor.y);  
  
  valor.y= 100.0;  
  printf("%s\n%s\n%s%d\n%s%f\n\n",  
         "Coloque um valor no membro de ponto flutuante",  
         "e imprima ambos os membros.",  
         "int:   ", valor.x,  
         "float  ", valor.y);  
  
  system("pause");  
}
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)



```

D:\Sergio\aula11b.exe
Coloque um valor no membro inteiro
e imprima ambos os membros.
int:    100
float   0.000000

Coloque um valor no membro de ponto flutuante
e imprima ambos os membros.
int:    1120403456
float   100.000000

Pressione qualquer tecla para continuar. . . _
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Operadores de Manipulação de Bits

- & - E bit a bit
- | - OU bit a bit
- ^ - OU exclusivo bit a bit
- << - deslocamento a esquerda
- >> - deslocamento a direita
- ~ - complemento de um

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Mascaramento

| Operação | Bits 7-0 | | | | | | | |
|----------|----------|---|---|---|---|---|---|---|
| A | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| B | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| A & B | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| A B | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |

Complemento de 1

A → 10010111

~A → 01101000

Deslocamento

A → 10010111

A >> 3 → 00010010 divisão

A << 2 → 01011100 multiplicação

```
#include<stdio.h>

main()
{system("color f0");
 unsigned x;
 void exibeBits(unsigned);

 printf("Digite um inteiro unsigned: ");
 scanf("%u", &x);
 exibeBits(x);

 printf("\n\nO numero dedeslocado 3 bits para a esquerda eh: \n");
 exibeBits(x << 3);
 printf("\n\n");
 system("pause");
}

void exibeBits(unsigned valor)
{
 unsigned c, exibeMascara = 1 << 15;

 printf("%7u = ", valor);
 for(c = 1; c <= 16; c++){
 putchar(valor & exibeMascara ? '1' : '0');
 valor <<= 1;

 if(c%8==0)
 putchar(' ');
 }
}
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

```
Digite um inteiro unsigned: 65000
65000 = 11111101 11101000

O numero dedeslocado 3 bits para a esquerda eh:
520000 = 11101111 01000000

Pressione qualquer tecla para continuar. . .
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Campos de Bits:

O C oferece a capacidade de especificar o número de bits no qual um membro `unsigned` ou `int` de uma estrutura ou união é armazenado.

Estabelecer o número de bits de um membro de uma estrutura ou união, do tipo `unsigned` ou `int`.

Permite melhor utilização da memória:

Usa-se :

Exemplo:

```
struct bitcarta {
    unsigned face: 4;
    unsigned naipe: 2;
    unsigned cor: 12;
};
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Constantes de Enumeração:

Constantes simbólicas cujos valores podem ser definidos automaticamente. Iniciam em zero (a menos que outro valor seja definido) e são incrementados de 1 em 1.

Exemplo:

```
enum meses {jan, fev, mar, abr, mai, jun};

enum meses2 {jul = 7, ago, set, out, nov, dez};
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Exemplo:

```
#include <stdio.h>

enum meses {JAN=1, FEV, MAR, ABR, MAI, JUN};

main ()
{
    enum meses mes;
    char*nomes[ ]={" ", "janeiro", "fevereiro",
                  "marco", "abril", "maio", "junho"};

    for (mes=JAN; mes<=JUN; mes++)
        printf ("%2d%11s\n", mes, nomes[mes]);

    system("pause");
}
```

JAN equivale ao numero 1, FEV ao numero 2, e assim por diante.

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

```
1      janeiro
2  fevereiro
3      marco
4      abril
5      maio
6      junho
Pressione qualquer tecla para continuar. . . _
```

Estruturas, Uniões, Manipulação de Bits e Enumeração (Cap.10)

Exercícios:

Pg 329, Fig. 10.2
Pg 337, Fig. 10.9
Pg 342, Fig. 10.16
Pg 345, Fig. 10.18