

2º Semestre

Técnicas de Programação

Luiz Fernando Carvalho

luizfcarvalhoo@gmail.com

Union

- Uma struct armazena, simultaneamente, todos os seus atributos:
 - É possível acessar cada um deles a qualquer momento, sem que isto modifique ou interfira nos demais atributos.
- Uniões (Union) também agregam atributos, mas se comportam de forma diferente;
- Uma união é um agrupamento de variáveis de tipos distintos, mas que **NÃO podem coexistir simultaneamente**;
 - Apenas um dos tipos pode estar armazenado na variável de cada vez.
- Como apenas uma variável é utilizada de cada vez, todas elas **compartilham o mesmo espaço na memória**;

Union

- Uma struct armazena, simultaneamente, todos os seus atributos:
 - É possível acessar cada um deles a qualquer momento, sem que isto modifique ou interfira nos demais atributos.
- Uniões (Union) também agregam atributos, mas se comportam de forma diferente;
- Uma união é um agrupamento de variáveis de tipos distintos, mas que **NÃO podem coexistir simultaneamente**;
 - Apenas um dos tipos pode estar armazenado na variável de cada vez.
- Como apenas uma variável é utilizada de cada vez, todas elas **compartilham o mesmo espaço na memória**;

Union

- A sintaxe de uma union é mostrada a seguir:

```
union Exemplo{  
    int i;  
    char c;  
};
```

- Como na definição de struct, o código acima não declara nenhuma variável, apenas define Exemplo como tipo union.
- Após sua definição, pode-se criar uma variável do tipo union:

```
union Exemplo v;
```

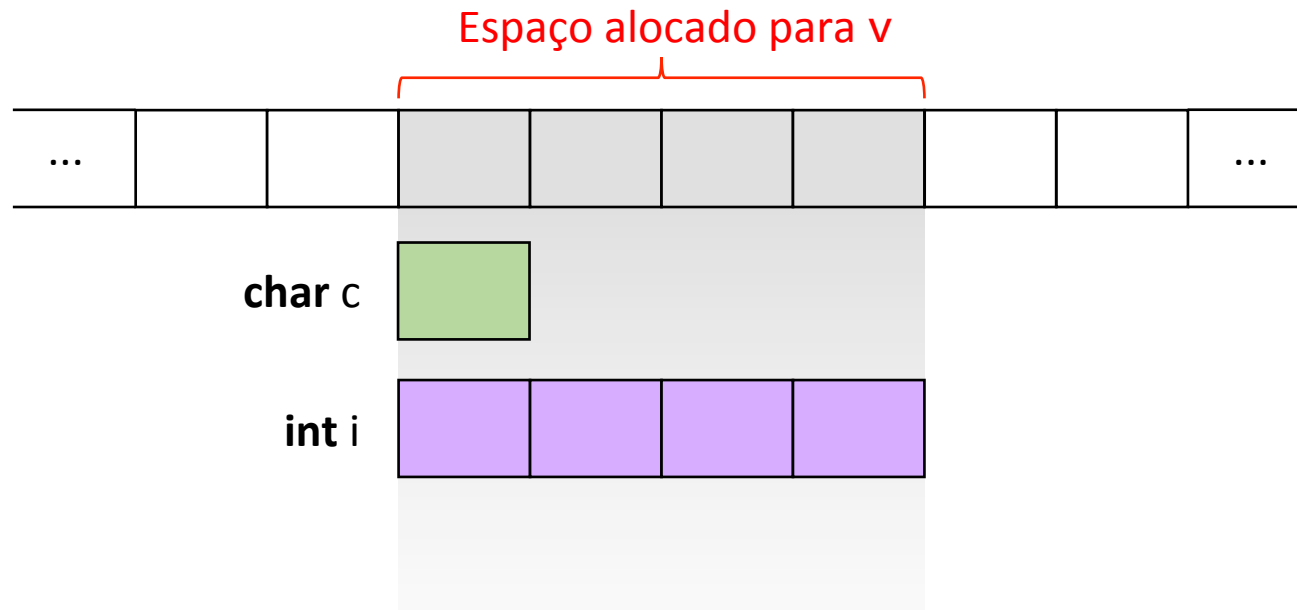
- Na variável v, os campos i e c compartilham o mesmo espaço de memória;

Qual é a quantidade de memória usada pela union Exemplo?

Union

```
union Exemplo{  
    int i;  
    char c;  
};
```

```
union Exemplo v;
```



O espaço usado pela union ocupa pelo menos o espaço necessário para armazenar o maior de seus campos

Union

```
union Exemplo{  
    int i;  
    char a, b, c, d;  
};
```

```
union Exemplo v;  
printf("%d", sizeof(v));
```

4 bytes

```
struct Exemplo{  
    int i;  
    char a, b, c, d;  
};
```

```
struct Exemplo v;  
printf("%d", sizeof(v));
```

8 bytes

1. Qual a diferença entre union e struct?
2. Quando podemos utilizar union?

Union

- Um uso interessante é quando não sabemos ao certo qual é o tipo de dados que uma variável irá armazenar;

```
union Numero{
    int i;
    double d;
};

int main(){
    union Numero n;

    n.d = 123456789.0;
    n.i = 15;

    printf("%f", n.d);
    printf("%d", n.i);

    ...
}
```

Importante: como `n.i` e `n.d` compartilham a mesma posição na memória, a atribuição de um valor ponto flutuante destrói o valor inteiro e vice versa!

→ 123456768.0
→ 15



Union

- Um uso interessante é quando não sabemos ao certo qual é o tipo de dados que uma variável irá armazenar;

```
union Caractere{  
    int i;  
    char c;  
};
```

```
int main(){  
    union Caractere n;
```

```
    n.c = 'a';
```

```
    n.i = 70;
```

```
    printf("%c", n.c);
```

```
    printf("%d", n.i);
```

```
    ...
```

F

70

Tabela ASCII (códigos de caracteres 0 - 127)

016 ►	032	048 0	064 @	080 P	096 `
017 ◀	033 !	049 1	065 A	081 Q	097 a
018 ‡	034 "	050 2	066 B	082 R	098 b
019 !!	035 #	051 3	067 C	083 S	099 c
020 ¶	036 \$	052 4	068 D	084 T	100 d
021 §	037 %	053 5	069 E	085 U	101 e
022 ■	038 &	054 6	070 F	086 V	102 f
023 ‡	039 '	055 7	071 G	087 W	103 g
024 ↑	040 (056 8	072 H	088 X	104 h
025 ↓	041)	057 9	073 I	089 Y	105 i
026 →	042 *	058 :	074 J	090 Z	106 j
027 ←	043 +	059 ;	075 K	091 [107 k
028 L	044 ,	060 <	076 L	092 \	108 l
029 ↔	045 -	061 =	077 M	093]	109 m
030 ▲	046 .	062 >	078 N	094 ^	110 n
031 ▼	047 /	063 ?	079 O	095 _	111 o

Enum

- **Definição 1:** É um tipo de dado o qual pode representar um único valor de um pequeno conjunto discreto e finito de alternativas;
- **Definição 2:** Um tipo de dado definido pelo usuário que define que uma variável poderá receber apenas um conjunto restrito de valores (constantes)
 - O valor de uma variável enumerada será sempre uma das opções dadas.
- *Por exemplo:* Para a escolha da forma de pagamento de um produto, o usuário pode escolher 3 opções: *dinheiro, cartão ou vale-refeição*;

```
char op;  
scanf("%c", &op);  
  
switch(op){  
    case 'd':  
        printf("Dinheiro..."); break;  
    case 'c':  
        printf("Cartao..."); break;  
    case 'v':  
        printf("Vale-refeição..."); break;
```

Enum

- *Por exemplo:* Para a escolha da forma de pagamento de um produto, o usuário pode escolher 3 opções: *dinheiro, cartão ou vale-refeição*;
- **Uma outra alternativa seria:**

```
int op;  
scanf("%d", &op);  
  
switch(op){  
    case 1:  
        printf("Dinheiro..."); break;  
    case 2:  
        printf("Cartao..."); break;  
    case 3:  
        printf("Vale-refeição..."); break;
```

Enum

- *Por exemplo:* Para a escolha da forma de pagamento de um produto, o usuário pode escolher 3 opções: *dinheiro, cartão ou vale-refeição*;
- **Uma terceira alternativa seria:**

```
#define DINHEIRO 1
#define CARTAO 2
#define VALE_REFEICAO 3

int main(){
    int op;
    scanf("%d", &op); //Valor 1, 2 ou 3

    switch(op){
        case DINHEIRO:
            printf("Dinheiro..."); break;
        case CARTAO:
            printf("Cartao..."); break;
        case VALE_REFEICAO:
            printf("Vale-refeição..."); break;
    }
```

Enum

- A sintaxe para se criar enumerações para o exemplo anterior é:

```
enum FormaPagamento {DINHEIRO, CARTAO, VALE_REFEICAO};  
  
int main(){  
    enum FormaPagamento op;    São constantes  
  
    op = DINHEIRO;  
  
    switch(op){  
        case DINHEIRO:  
            ...  
        case CARTAO:  
            ...  
        case VALE_REFEICAO:  
            ...  
        default:  
            ...  
    }  
    ...  
}
```

Enum

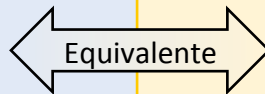
- A sintaxe para se criar enumerações para o exemplo anterior é:

```
enum FormaPagamento {DINHEIRO, CARTAO, VALE_REFEICAO};

int main(){
    enum FormaPagamento op;

    op = dinheiro;

    switch(op){
        case DINHEIRO:
            ...
        case CARTAO:
            ...
        case VALE_REFEICAO:
            ...
        default:
            ...
    }
    ...
}
```



```
op = dinheiro;

switch(op){
    case 0:
        ...
    case 1:
        ...
    case 2:
        ...
    default:
        ...
}
...
```

Importante!

Automaticamente o compilador lista as constantes como um valor **inteiro**, isto é:

DINHEIRO = 0

CARTAO = 1

VALE_REFEICAO = 2

Enum

- Outro exemplo:

```
enum Semana {SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};

int main(){
    enum Semana hoje;

    hoje = QUINTA;

    printf("Valor = %d", hoje);

    return 0;
}
```

Saída:

Valor = 3

Enum

- Olhando mais de perto... enum declara nomes para valores inteiros

```
enum Semana {SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};
```

0 1 2 3 4 5 6

- Se começarmos enumerar a partir do 1, temos:

```
enum Semana {SEGUNDA=1, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};
```

1 2 3 4 5 6 7

- Podemos colocar quaisquer valores:

```
enum Semana {SEGUNDA=150, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};
```

150 151 152 153 154 155 156

```
enum Semana {SEGUNDA=1, TERCA, QUARTA=8, QUINTA, SEXTA, SÁBADO=49, DOMINGO};
```

1 2 8 9 10 49 50

Enum

- Outro exemplo:

```
enum Semana {SEGUNDA, TERCA, QUARTA, QUINTA, SEXTA, SABADO, DOMINGO};

int main(){
    enum Semana hoje;

    hoje = QUINTA;

    if(hoje == SEGUNDA)
        ...
    else if(hoje == TERCA)
        ...
    else if(hoje == QUARTA)
        ...

    return 0;
}
```


Enum

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

```
/*
```

```
0    BLACK
```

```
1    BLUE
```

```
2    GREEN
```

```
3    CYAN
```

```
4    RED
```

```
5    MAGENTA
```

```
6    BROWN
```

```
7    LIGHTGRAY
```

```
8    DARKGRAY
```

```
9    LIGHTBLUE
```

```
10   LIGHTGREEN
```

```
11   LIGHTCYAN
```

```
12   LIGHTRED
```

```
13   LIGHTMAGENTA
```

```
14   YELLOW
```

```
15   WHITE*/
```

```
enum Cores {PRETO, AZUL, VERDE, CIANO, VERMELHO, MAGENTA,
MARROM, CINZA_CLARO, CINZA_ESCURO, AZUL_CLARO, VERDE_CLARO,
CIANO_CLARO, VERMELHO_CLARO, MAGENTA_CLARO, AMARELO, BRANCO};
```

```
int main(){
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    enum Cores cor;

    cor = VERMELHO;

    SetConsoleTextAttribute(hConsole, cor);
    printf("TEXTO COLORIDO!!!\n");

    return 0;
}
```

Exemplo usando Struct e Enum

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<windows.h>
4
5  #define MAX 16
6  typedef struct{
7      int x, y;
8  }Posicao;
9
10 enum Movimento {CIMA = 'w', BAIXO = 's', ESQUERDA = 'a', DIREITA = 'd'};
11 void iniciaMapa(char m[MAX][MAX]){
12     int i, j;
13
14     for(i=0;i<MAX;i++){
15         for(j=0;j<MAX;j++){
16             m[i][j] = ' ';
17         }
18         for(i=0;i<MAX;i++)
19         {
20             m[i][0] = 'x';
21             m[i][MAX-1] = 'x';
22         }
23         for(j=0;j<MAX;j++){
24             {
25                 m[0][j] = 'x';
26                 m[MAX-1][j] = 'x';
27             }
28         }
```

Exemplo usando Struct e Enum

```
29 void imprime(char m[MAX][MAX]){
30     int i, j;
31
32     system("cls"); //LIMPA A TELA
33     for(i=0;i<MAX;i++)
34     {
35         for(j=0;j<MAX;j++)
36             printf("%c", m[i][j]);
37         printf("\n");
38     }
39 }
40 int main(){
41     char mapa[MAX][MAX], letra = 'u';
42     Posicao player = {7,5};
43
44     iniciaMapa(mapa);
45     mapa[player.x][player.y] = '@';
46     imprime(mapa);
47
48     while(letra != 'q'){
49         letra = getch();
50
51         if(letra == CIMA){
52             mapa[player.x][player.y] = ' ';
53             player.x--;
54             mapa[player.x][player.y] = '@';
55         }
56         ...
57     }
```