PROGRAMAÇÃO DE COMPUTADORES V - TCC- 00.323

Modulo 10 : Tipos especiais de dados: **união, enumeração e ...**

Aura - Erick
aconci@ic.uff.br, erickr@id.uff.br

Roteiro

- **Union**
- **Enumerações**
- **Goto , label**
- **exercicios**

Motivação - **Union**

- Armazenar diferentes tipos de dado em uma mesma área de memória

- Ver o mesmo valor como inteiro, real, caractere, booleno, etc.

que é union ?

- A **união/ union** é um tipo de dado disponíveis em C que permite armazenar diferentes tipos de dados no mesmo local de memória.

- Você pode definir uma **union** / **união** com muitos membros, mas apenas um membro pode conter, o valor a um dado qualquer momento.

- **union** oferece uma maneira eficiente de usar os mesmos locais de memória para múltiplo-propósito.

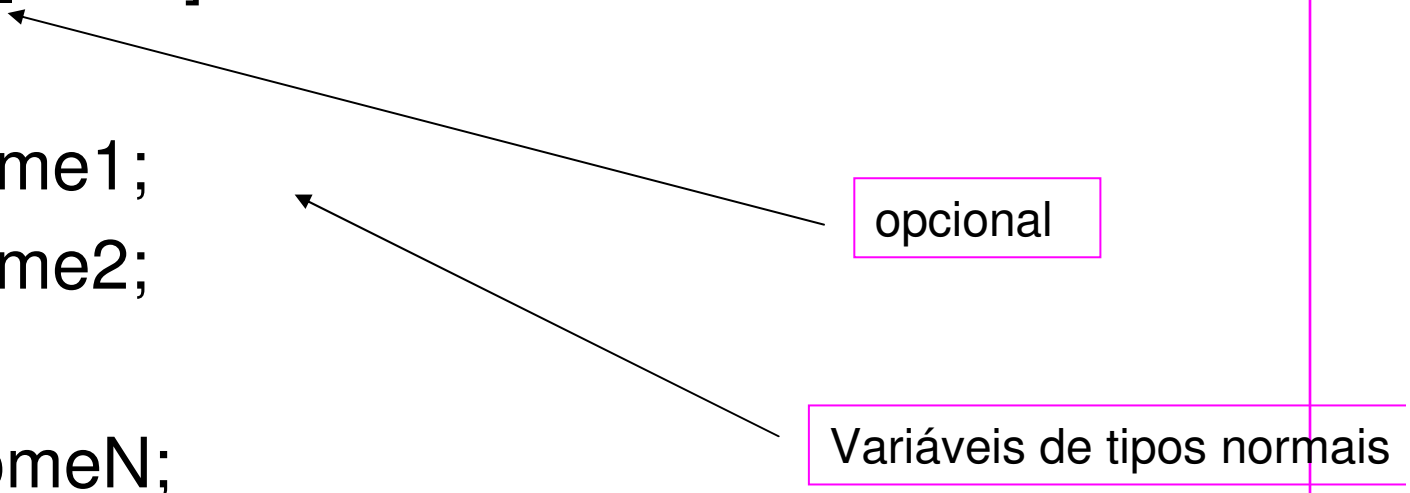
Definição de uma União - **Union**

- Para definir uma **união**, você deve usar a instrução **union** , da mesma forma como você fez para a definição de uma estrutura.

- A declaração da **union** / **união** define um novo tipo de dados com mais de um membro para o seu programa.

Union - prototipo

```
union [union_nome]
   {
    tipo_1   nome1;
    tipo_2   nome2;
    ...
    tipo_N   nomeN;
    } [nome_de_variaveis];
```



Union - exemplo

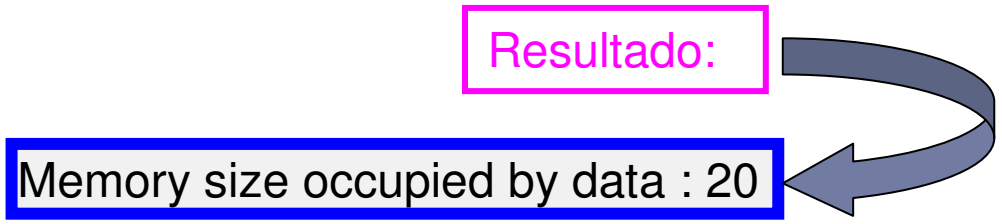
- uma **union** de nome **Data** com 3 membros i, f, e str

```
union Data
  {
   int i;
   float f;
   char str[20];
   }  data1, data2;
```

Espaço ocupado pela variável

- A memória ocupada por uma união será grande o suficiente para caber o maior membro da união.
- Por exemplo, no exemplo anterioir,a **Union** Data ocupará 20 bytes de memória pois este é o espaço que será ocupado pela cadeia de caracteres.
- O exemplo a seguir mostra como ver o tamanho da memória total ocupado pela **Union -** união

```
#include <stdio.h>
#include <string.h>
union Data {
   int i;
   float f;
   char str[20];
};
int main( ) {
union Data data;
printf( "Memoria ocupada: %d\n", sizeof(data));
return 0;
}
```



Usando os elementos da Union//Accessing Union Members

- Para usar os membros da **union,** se usa o operador **ponto: .** .
- A palavra-chave **union** permite você declarar as variáveis de um tipo definido antes.
- the keyword **union** is used to define variables of union type.
- Vejamos mais um exemplo de programa :

```
#include <stdio.h>
#include <string.h>
union Data {
   int i;
   float f;
   char str[20];};

int main( )
{
union Data data;
data.i = 10;
data.f = 220.5;
strcpy( data.str, "C Programming");

printf( "data.i : %d\n", data.i);
printf( "data.f : %f\n", data.f);
printf( "data.str : %s\n", data.str);
return 0
;}
```

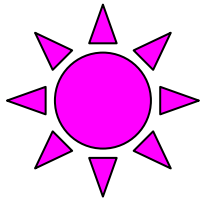
Quando o código acima é compilado e executado, ele produz o resultado :

```
data.i : 1917853763
data.f : 412236058032779486045275994368.000000
data.str : C Programming
```

valores de **i** e **f** foram **corrompidos** porque o valor final atribuído à variável ocupou a **mesma posição de memória** e por isso o valor de **str** está sendo impressa muito bem.

Mas **não acredite** em mim!!!

- Rode para ver se está correto!!!



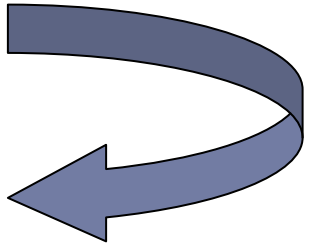
Agora vamos olhar para o mesmo exemplo,
mais uma vez!

Mas agora em uma aplicação com mais sentido!

```
#include <stdio.h>
#include <string.h>
union Data
{
  int i;
  float f;
  char str[20];
};

int main( )
{
union Data data;
data.i = 10;
printf( "data.i : %d\n", data.i);
data.f = 220.5;
printf( "data.f : %f\n", data.f);
strcpy( data.str, "C Programming");
printf( "data.str : %s\n", data.str);
return 0;
}
```

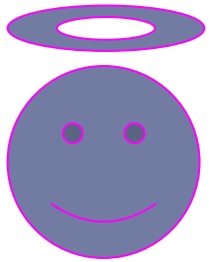
```
data.i : 10
data.f : 220.500000
data.str : C Programming
```



todos **estão sendo impressos** muito bem porque só um membro está sendo usado de cada vez.

E a segunda

- Atração do dia!!!!



Enumeration - enumeração

- Uma enumeração é um tipo de dados **definido pelo usuário .**
- Consiste de constantes inteiras cada uma com um nome.
- **enum** é a palavra-chave é usada para definir esse tipo de dados.

- An enumeration is a user-defined data type, it consists of integral constants and each integral constant is give a name.
- Keyword **enum** is used to defined this data type.

Protótipo:

enum **type_name** { value1, value2,...,valueN };

- Aqui, **type_name** é o nome do tipo de dados enumerado.
- E **value1, value2** ...., são os valores do tipo **type_name.**

- Por **default**, **valu**e1 será igual a **0**, **value2** será de **1** e assim por diante, mas, o programador pode alterar esse valor padrão.

Exemplo: valores dos naipes:

```
enum naipesBaralho
{
paus=0;
ouro=10;
copas=20;
espadas=3;
};
```

Meses do ano:

```
enum mesesEmPortuguesnaipesBaralho
{
janeiro=1;
outubro=10;
marco=3;
};
```

Outro mais:

```
#include <stdio.h>
enum week{sunday,monday,tuesday,wednesday,thursday,friday,saturday};
int main()
{
enum week today;
today=wednesday;
printf("%d day",today+1);
return 0;
}
```

Saida : 4 day

Saida : 2 a-feira

```
#include <stdio.h>
enum dias{sabado,domingo,segunda,terça,quarta,quinta,sexta};
int main()
{
enum dias hoje;
hoje=segunda;
printf("%d a-feira",hoje);
return 0;
}
```

Mas não acredite em mim!!!

Rode para ver se esta correto!!!

Obs:

- Você pode escrever programas em C, sem a ajuda de **Enumerações**, mas **enum** ajuda seus códigos a ficarem mais claros e simples.

- You can write any program in C language without the help of **enumerations** but, **enum** helps in writing clear codes and simplify programming.

goto

- A instrução **goto** em C fornece um **salto incondicional** do ‘goto' para uma instrução **rotulada** na mesma função.

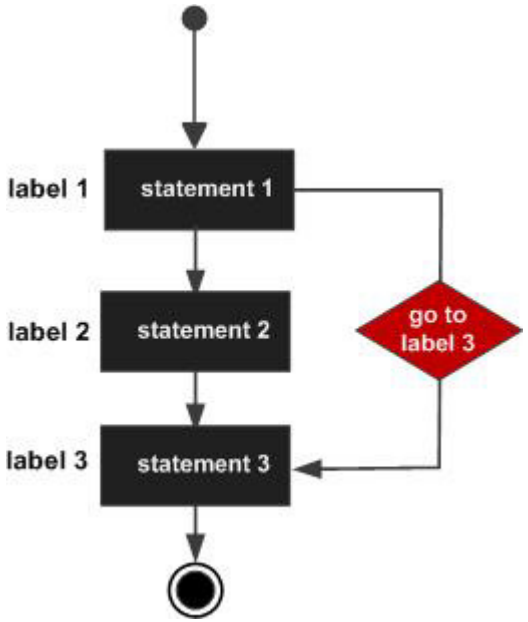
- NOTA - Utilização de instrução **goto** é altamente **desaconselhavel em qualquer linguagem de programação**, porque torna difícil rastrear o fluxo de um programa, fazendo o programa difícil de se entender e difícil de se modificar.
- Qualquer programa a usa um **goto** pode ser reescrito para evitar seu uso!!!.

Syntax

goto label;

.. .

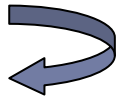
label: instruções;



exemplo

```
#include <stdio.h>
int main () {
   int a = 10;
   /*  label */
   LOOP: do {
         if( a == 15) {
               /*vai pular isso */
               a = a + 1;
               goto LOOP;
          }
          printf("valor de a: %d\n", a);
          a++;
     } while( a < 20 );
     return 0;
}
```

resultado



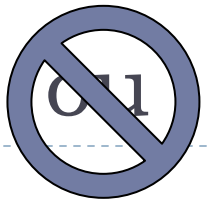
```
valor de a: 10
valor de a: 11
valor de a: 12
valor de a: 13
valor de a: 14
valor de a: 16
valor de a: 17
valor de a: 18
valor de a: 19
```

Quinto(A) Trabalho

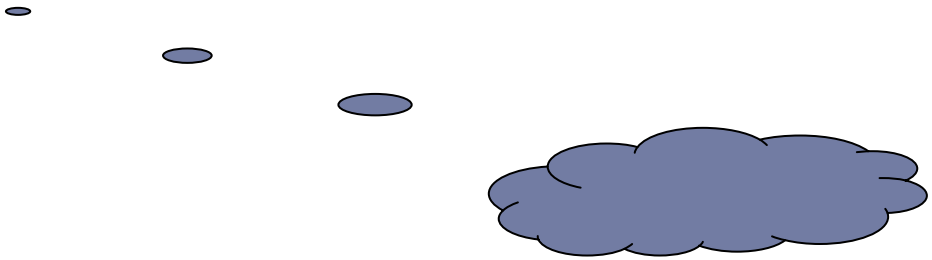
- **Re escreva o Quinto Trabalho** para que os meses do nascimento e atual sejam escritos por extenso e definidos por uma enumeração.
- Use essa enumeração para lhe ajudar a calcular a idade da pessoa!

“uma ideia” pensamos em “voce sugerir”!

- Você propor o trabalho que use o conceito de “**estruturas**”
- Baseado em um controle de estoque/produção e custos.
- De modo que você pense em uma firma que produz algum item (por exemplo peças de carros, equipamentos de industria alimentícia, confecção, etc) e você deve propor um programa que ajude na manutenção do estoque e lucro.
- Na estrutura deve haver pelo menos coisas como: nome da peça, número, custos, validade.
- O que acham? Funciona?



- Teria algum outro programa que voce gostaria de propor que usasse nossos conceitos ?
- A hora é agora!!!
- De sua idéia!!



Referencias

http://www.tutorialspoint.com/cprogramming/c_unions.htm

http://www.ime.usp.br/~pf/algoritmos/

http://www.programiz.com/c-programming/c-enumeration

http://www.paulotrentin.com.br/programacao/curso-gratuito-de-programacao-em-c/aula-7-estruturas-unioes-enumeracoes-e-tipos-definidos-pelo-usuario/

http://wiki.icmc.usp.br/images/6/6b/Revis%C3%A3o-Estruturas.pdf

http://www.numaboa.com.br/informatica/tutos/c/1004-c-estruturas

https://pt.wikibooks.org/wiki/Programar_em_C/Uni%C3%A3o

http://equipe.nce.ufrj.br/adriano/c/apostila/estru.htm