C
Tipos de dados avançados

Uma visao geral

http://nataniel.tk

Structs
Unions
Enums

Structs

- Usado em programas de media e grande dimensao.
- Une um conjunto de tipos de dados num novo tipo de dado.
- Representação em memoria.
- Como qualquer tipo de dado pode ser usado normalmente.
- Alguns cuidados especiais na manipulação com ponteiros.

Structs

- Imagine um programa aonde tenha que declarar 10 ou mais variaveis e manipula-las enviando-as a funcões.

```
int a, b, c;
unsigned long int d;
float sal, debito;
char e, f[], *g;
...
```

Structs

- Declaração:

```
struct nomedotipo{
membros
..
}variaveis;
```

Structs

- Ou:
- Declaração:

```
struct nomedotipo{
membros
..
};
struct  nomedotipo var1, var2, var3[10];
```

Structs

- Exemplo:

1 - Declare uma estrutura capaz de suportar a definição dum novo tipo bilhetedeidentidade com todos os seus membros.

R: Sabemos que um B.I possui alguns campos como: Nome, filiação, numero de bilhete, altura, sexo, validade etc.

Structs

```
struct bilhetedeidentidade{
char nome[150];
char filia_pai[150];
char filia_mae[150];
char bi[20];
char sexo;
float altura;
int dia_exp;
int mes_exp;
int ano_exp;
};
```

Structs

- Como declarar variaveis para esse novo tipo de dados?

R: 2 formas:

Structs

- A)

```
struct bilhetedeidentidade{
char nome[150];
char filia_pai[150];
char filia_mae[150];
char bi[20];
char sexo;
float altura;
int dia_exp;
int mes_exp;
Int ano_exp;
}var1, var2, var[10];
/* variavel 3 com 10 elementos */
```

Structs

- B)

```
struct bilhetedeidentidade{
char nome[150];
char filia_pai[150];
char filia_mae[150];
char bi[20];
char sexo;
float altura;
int dia_exp;
int mes_exp;
int ano_exp;
};
struct bilhetedeidentidade var1, var2, var3[10];
```

Structs

- Que tipos representam estas variaveis em memoria?

R: O tipo struct bilhetedeidentidade

A terceira variavel (var3[10]) representa o tipo indicado acima com 10 elementos.

Structs

- Como acessar membros do tipo?

Pela notação variavel.membro;

- Ex: Acessar variavel 1 e inserir um nome;

```
char  name = “nataniel”;
strcpy(var1.nome, name);
```

Ex: Acessar variavel 1 e inserir altura e sexo:

```
var1.altura = 1.60;
var1.sexo = ‘m’;
```

Structs

- Acessar *2* elemento da *var3* e inserir filiacao do pai.

strcpy(var3[1].filia_pai, “joaquinito da costa”);

Structs

- Typedefs também se aplicam a structs

Forma geral do typedef:

typedef tipo_conhecido novo_tipo;

Forma geral em estruturas:

typedef struct tipo{

membros

...

}**identificador**;

identificador var1, var2, var3[10];

Structs

```
typedef struct bilhetedeidentidade{
char nome[150];
char filia_pai[150];
char filia_mae[150];
char bi[20];
char sexo;
float altura;
int dia_exp;
int mes_exp;
int ano_exp;
}BI;
BI var1, var2, var3[10];
```

Structs

- Possivel estruturas de estruturas. Como?

No nosso exemplo poderiamos separar os membros da data de expiração numa nova estrutura, desde que esteja antes da estrutura mae ou principal.

Structs

- Formato geral:

```
typedef struct{
membros_filho;
...
}identificador_filho;

typedef struct nome{
identificador_filho membro_do_membro_filho;
...
}identificador_mae;
```

Structs

- Acesso também através da notação pontual.

Ex:

```
typedef struct{
int dia_exp;
int mes_exp;
int ano_exp;
}dataexpiracao;

typedef struct bilhetedeidentidade{
char nome[150];
char filia_pai[150];
char filia_mae[150];
char bi[20];
char sexo;
float altura;
dataexpiracao data;
}BI;
BI var1, var2, var3[10];
```

Structs

- Acessar e inicializar membro ano_exp de *var1*

```
var1.data.ano_exp = 2011;
```

- Acessar e inicializar membro dia_exp do *4º* elemento de *var3*

```
var3[3].data.dia_exp = 12;
```

Structs

- Passagem de estruturas a funções

Passa-se o tipo ou o typedef

nome_funcao(tipo ou typedef);

Ex: Passar estructura a funcao para alterar imprimir nome do bilhete de identidade da primeira variavel.

Structs

Pelo typedef

void imprime(BI x);

{ … .. }

Ou pelo tipo

void imprime(struct bilhetedeidentidade x);

{ … .. }

Structs

- Declarar e inicializar nova struct:

```
BI new_var;
new_var.nome = “nataniel”;
new_var.sexo = ‘m’;
new_var.bi = “737j2sjsjmsmdmdsx”;
....
```

Ou

```
BI new_var = {“nataniel”, “joaquinito da costa”,
    “joaquinita da costa”, “737j2sjsjmsmdmdsx”, ‘m’,
    1.60, {17, 12, 2011}};
```

Structs

- Passar struct x a funcao imprime

```
void imprime(struct bilhetedeidentidade x)
{
   printf("%s\n", x.nome);
   printf("%c\n" x.sexo);
   printf("%s\n" x.bi);
}
```

Structs

- Como alterar dados desta estrutura?
- Precisamos do endereço da estrutura.

Passando um endereço de memoria da estrutura para a função conseguimos acesso directo aos seus membros podendo manipula-los pela memoria.

```
altera(&x);
```

Structs

```
void altera(BI *x)
{  printf("Digite nome :");
   gets((*x).nome);
   printf("Digite numero de bi :");
   gets((*x).bi);
   printf("Digite o sexo :");
   scanf("%c", &(*x).sexo);
   ...
   }
```

Structs

- O que acontece?

scanf(“%s”, (*x).nome); ??????

R: - Recebemos o endereço da estrutura, armazenamos numa variavel apontadora.
- X = &x[0];
- Logo se x fosse uma *estrutura* armazenavamos usando x.nome.
Mas como é um *apontador para uma estrutura* usamos (*x.nome) que nao é correcto usar ja que desejamos o endereço de x e como ‘.’ tem maior precedencia que ‘*’ logo o compilador analisará primeiro interpretará mal o argumento, logo envolvemos *.x em parenteses.

Unioes

- Pouca usabilidade.
- Também cria novo tipo.
- Geralmente juntamente com Structs.
- Similaridade com Structs.
- Diferença no tamanho de dados sendo *Record Variant.*
- O tamanho da Uniao é o tamanho do tipo que ocupa maior espaço em memoria.

Unioes

```
union bilhete{
Int idade;
float altura;
};
union bilhete do_manuelito;
do_manuelito.idade = 40;
do_manuelito.altura = 1.80;
```

Unioes

- Tamanho:

Usando operador sizeof();

R: sizeof(union bilhete);

Verá que retorna tamanho do tipo float o maior tipo presente na uniao.

Se alterado valor de um dos membros da uniao todos os conteudos da mesma uniao serao alterados.

Enums

- Enumerações como o nome diz
- A cada membro duma Enum é atribuido um valor por defeito.
- Inicia com 0 se nao definido qualquer valor.
- Se nao definido um valor num membro, este procura adicionar 1 ao valor do membro anterior.

Enums

```
enum vogais{a, e, i, o, u};
void main()
{  int var1 = e;
   if(a)
        printf(“verdadeiro”);
  printf(“\nFalso. Verdadeiro é: %d”, var1);
   }
```

Enums

```
enum vogais{ a=1, e, i=20, o, u};
```

Que valor passam a ter e, o, u?

e=2

o=21

u=22

Conclusões

- Maior usabilidade e facilidade ao programador.
- Importantissimo para medios a grandes programas.

Agradecimentos

- A deus pela inteligencia

Who i am?

- Angolana PERL Monger mantenedor. (angola.pm.org)
- Ticaquista (ticaoc.cjb.net)
- Blogger (nataniel.tk)
- Computer Networks and Telecommunication Student at UCAN/Angola.