# MC 102 turma Z - 2sem2012
# Linguagem C
# Referências

Prof. Fernando Vanini

IC - Unicamp

# Referências (ou endereços)

- Todo acesso a variáveis e funções por um programa é feito com base em endereços de memória ou referências.
- Numa linguagem como C, o uso desses endereços normalmente fica restrito ao código gerado pelo compilador. O programador não precisa saber da sua existência.
- Em alguns casos no entanto, o programador precisa recorrer aos endereços para obter o efeito desejado. Um exemplo:

```
int n; float x;
…
scanf("%d %f", &n, &x);
```

- Nesse exemplo, para que a função scanf() altere os valores de n e x, foi necessário usar o operador '&', que retorna o endereço da variável à qual é aplicado.

# O operador ‘&’

- O operador ‘&’ (‘operador de endereço’) retorna o endereço de memória da variável à qual é aplicado.
- Um dos usos do operador ‘&’ é na passagem de parâmetros para funções que esperam um endereço de memória como parâmetro, ou *parâmetros passados por referência*’ na terminologia de C.
- A função scanf(), usada no exemplo anterior, é uma função na qual os parâmetros são passados por referência.

# Passagem de parâmetros por referência

- Ao declarar uma função, é possível indicar que um parâmetro é passado por referência através de um 'modificador' de tipo do parâmetro, representado pelo caracter '*'. Um exemplo:

```
void xbluft(int *k){
      k = (k / 100)*100;
}
```

- O exemplo define a função xbluft() como tendo um parâmetro do tipo 'endereço de inteiro'.
- Importante: o caracter '*' tem significados diferentes, em função do contexto em que é usado. O exemplo ilustra os dois tipos de uso: como 'modificador de tipo', e também como operador de multiplicação.
- Ao usar a função xbluft(), o valor passado como parâmetro deve ser o endereço de uma variável inteira. Um exemplo:

```
int n = 91181;
printf("%d\",xbluft(&n));   // imprime 91100
```

# O ‘tipo endereço’

- Através do uso do modificador de tipos, é possível declarar variáveis do tipo endereço (ou *apontadores*, na terminologia C).

Um exemplo:

```
int n = 91181;
int *p = &n;
xbluft(p);
printf(“%d %d,*p,n);
```

*No exemplo a variável p é do tipo ‘endereço de inteiro’ ou ‘apontador para inteiro’. Como o tipo de p é o mesmo que o tipo do parâmetro da função xbluft(), p pode ser passado diretamente como parâmetro para essa função. É importante notar que p e &n se referem à mesma posição de memória.*

# Acesso ao valor a partir de um endereço

Para acesso ao valor a partir de um endereço, usa-se o operador '*' antes do nome da variável que contém o endereço. Essa forma de acesso foi mostrada no exemplo anterior:

```
int n = 91181;
int *p = &n;
xbluft(p);
printf("%d %d,*p,n);
```

*No exemplo*

*int *p;*

*declara p como um apontador para inteiro. O valor apontado por p é acessado numa expressão através da operação *p.*

*Poderíamos fazer, por exemplo*

*int x = (*p) + 32;*

# Um exemplo

A função abaixo troca o valor de duas variáveis. Ela ilustra o uso de parâmetros passados por referência e o acesso ao valor referenciado por um apontador:

```
void troca(int *a, int *b){
      int t = *a;
      *a = *b;
      *b = t;
}
```

*Nesse exemplo, a variável t, inteira, tem como valor inicial o valor apontado por a.*

- Exemplo de uso:

```
int x = 10, y = 20;
troca(&x,&y);
```

# Vetores como parâmetro

- Um vetor passado como parâmetro é implicitamente passado por referência. Exemplo:

```
void ordena(int v[], int n){
  int i,b;
  do{
     b = FALSE;
     for(i = 0; i < (N-1); i++){
        if(v[i] > v[i+1]) {
           troca(&(v[i]),&(v[i+1]));
           b = TRUE;
        }
     }
  } while(b);
}
```

O vetor v é passado por referência:
- o acesso aos elementos de v não precisa do modificador '*'.
- Qualquer alteração de valor de um elemento de v é uma alteração no valor do vetor 'original'.
- Notar o uso do operador '&' na chamada a troca().

# Vetores como parâmetro

- Exemplo de uso da função ordena():

```
int main(){
    int w[] = { 3, 5, 2, 7, 1, 8, 9, 4 };
    ordena(w,8);
    int i;
    for(i = 0; i < N; i++)
        printf("w[%d]:%d\n",i,w[i]);
}
```

# Vetores e Apontadores

- O tipo vetor é compatível com o tipo apontador. Uma variável do tipo vetor de inteiros pode ser declarada como um apontador para inteiro. Exemplo:

```
int main(){
    int *w = { 3, 5, 2, 7, 1, 8, 9, 4 };
    ordena(w,8);
    int i;
    for(i = 0; i < N; i++)
        printf("w[%d]:%d\n",i,w[i]);
}
```

# Vetores e apontadores

- A função ordena() mostrada anteriormetne, pode ser definida da seguinte forma:

```
void ordena(int *v, int n){
  int i,b;
  do{
     b = FALSE;
     for(i = 0; i < (N-1); i++){
        if(v[i] > v[i+1]) {
           troca(&(v[i]),&(v[i+1]));
           b = TRUE;
        }
     }
  } while(b);
}
```

# Strings e apontadores

- O tipo string, que é compatível com vetor de caracteres, também é compativel com o tipo ‘apontador para caracter’, conforme mostra o exemplo a seguir:

```
char *str = “um string exemplo”;
```