Projeto e Análise de Algoritmos
Linguagem C - Parte 2

Antonio Luiz Basile

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie

February 15, 2018

Laço via recursão

- Um programa recursivo bem definido chama a si mesmo, direta ou indiretamente, numa instância menor.
- Deve também ter uma condição de parada, ou seja, um caso base.

```
int fat (int n)
{
   if (n == 0) return 1;
   else return n * fat(n-1);
}
```

"Para entender recursão,
é preciso entender recursão."

Laço via recursão (continuação)

```
int maxRec (int n, int v[])
{
    if (n == 1)
        return v[0];
    else {
        int x;
        x = maxRec (n-1, v);  // x é o máximo de v[0..n-2]
        if (x > v[n-1]) return x;
        else return v[n-1];
    }
}
```

Ponteiros

```
int x = 1, y = 2, z[10];
int *ip;          /* ip is a pointer to int */

ip = &x;          /* ip now points to x */
y = *ip;          /* y is now 1 */
*ip = 0;          /* x is now 0 */
ip = &z[0];       /* ip now points to z[0] */
```

Figure: Ponteiros (K&R)

Passagem por Valor

```c
void troca (int a, int b){
    int aux;
    aux = a;
    a = b;
    b = aux;
}
int main (void){
    int x = 1, y = 2;
    printf ("Antes: x = %d e y = %d     ", x, y);
    troca (x, y);
    printf ("Depois: x = %d e y = %d\n", x, y);
    return 0;
}
```

Antes: x = 1 e y = 2 Depois: x = 1 e y = 2

Passagem por Referência

```c
void troca (int *a, int *b){
    int aux;
    aux = *a;
    *a = *b;
    *b = aux;
}
int main (void){
    int x = 1, y = 2;
    printf ("Antes: x = %d e y = %d     ", x, y);
    troca (&x, &y);
    printf ("Depois: x = %d e y = %d\n", x, y);
    return 0;
}
```

Antes: x = 1 e y = 2 Depois: x = 2 e y = 1

Ponteiros e Vetores

```
int a[10];
```

defines an array `a` of size 10, that is, a block of 10 consecutive objects named `a[0]`, `a[1]`, ..., `a[9]`.

The notation `a[i]` refers to the `i`-th element of the array. If `pa` is a pointer to an integer, declared as

```
int *pa;
```

then the assignment

```
pa = &a[0];
```

sets `pa` to point to element zero of `a`; that is, `pa` contains the address of `a[0]`.

Figure: Ponteiros e Vetores (K&R)

Ponteiros e Vetores

Now the assignment

```
x = *pa;
```

will copy the contents of `a[0]` into `x`.

If `pa` points to a particular element of an array, then by definition `pa+1` points to the next element, `pa+i` points `i` elements after `pa`, and `pa-i` points `i` elements before. Thus, if `pa` points to `a[0]`,

```
*(pa+1)
```

refers to the contents of `a[1]`, `pa+i` is the address of `a[i]`, and `*(pa+i)` is the contents of `a[i]`.

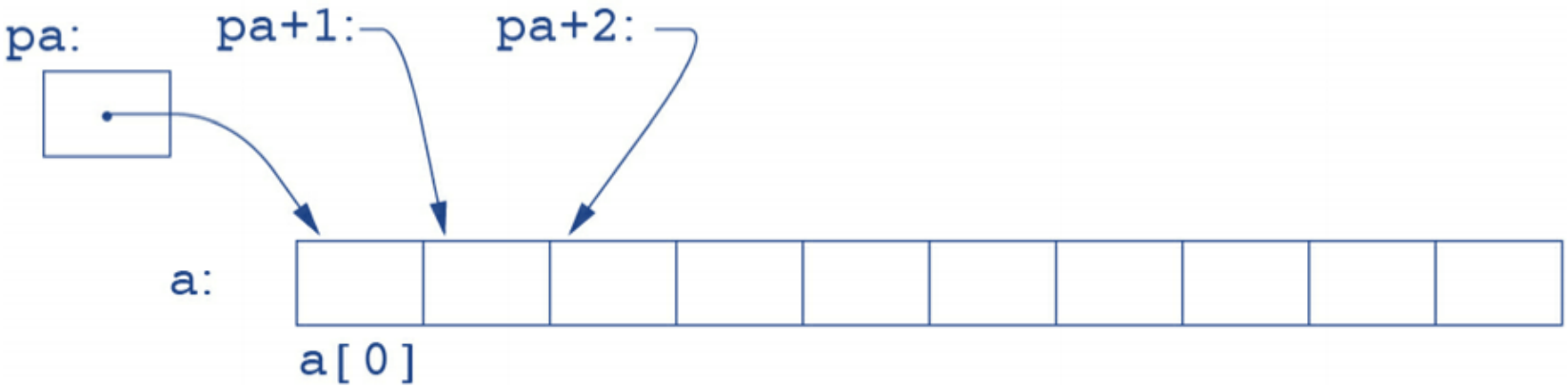

Figure: Ponteiros e Vetores (K&R)

Alocação Dinâmica de Vetores bidimensionais

```
int **malloc2d (int lin, int col)
{
    int i;
    int **t = malloc (lin * sizeof (int *));
    for (i = 0; i < lin; i++)
        t[i] = malloc (col * sizeof (int));
    return t;
}

int **a = malloc2d (3, 5);
```