

Ponteiros em Pascal

Introdução à Ciência da Computação

Rosane Minghim

Apoio na confecção: Rogério Eduardo Garcia

Danilo Medeiros Eler

Ponteiros em Pascal: Declaração e operadores

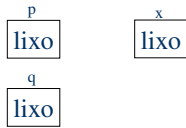
- Em Pascal, o funcionamento de ponteiros é parecido com o do pseudocódigo.
- Declaração:
var variável: ^tipo;
- Operador de endereço: @
- Ponteiro nulo: nil;
- Operador de conteúdo: ^

Ponteiros em Pascal

Exemplo

```
var
  x: integer;
  p, q: ^integer;

x := 10;
p := @x;
q^ := 30; { ERRO: q aponta p/ lixo }
q := p;
q^ := p^ + 10;
writeln(p^);
```



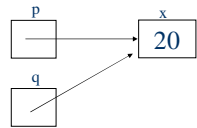
Ponteiros em Pascal

Exemplo

```
var
  x: integer;
  p, q: ^integer;

x := 10;
p := @x;

q := p;
q^ := p^ + 10;
writeln('Resultado = ', p^);
```



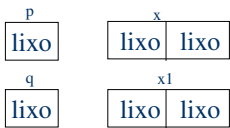
Resultado = 20

Ponteiros em Pascal

Exemplo 2

```
type
  pont_rec = ^rec;
  rec = record
    dado: real;
    pont: pont_rec;
  end;

var
  x, x1: rec;
  p, q: pont_rec;
```

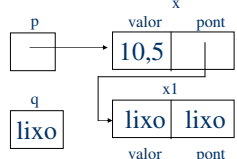


Ponteiros em Pascal

Exemplo 2

```
var
  x, x1: rec;
  p, q: pont_rec;

x.valor := 10.5;
p := @x;
p.pont := @x1;
```

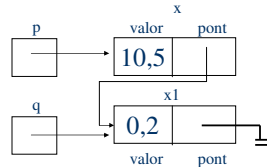


Ponteiros em Pascal

Exemplo 2

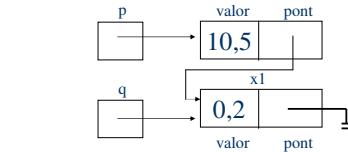
```
var
  x,x1:rec;
  p,q :pont_rec;

  x.valor := 10.5;
  p := @x;
  x.pont := @x1;
  x1.valor := 0.2;
  q := x.pont;
  x1.pont := nil;
```



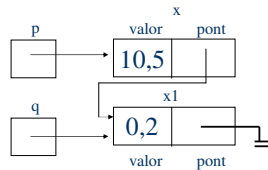
Ponteiros em Pascal

Exemplo 2



```
writeln(p^.valor)
writeln(q^.valor)
writeln(p^.pont^.valor)
writeln(x.valor)
writeln(x.pont^.valor)
writeln(x1.valor)
```

Ponteiros em Pascal – Exemplo 2



```
writeln(p^.valor)
writeln(q^.valor)
writeln(p^.pont^.valor)
writeln(x.valor)
writeln(x.pont^.valor)
writeln(x1.valor)
```

10.5
0.2
0.2
10.5
0.2
0.2

Uso de Ponteiros para Alocação dinâmica – Pascal

Alocando espaço na Heap

- Em Pascal, a operação que aloca memória é implementada por uma função, na forma:

```
new(p);
```

que tem o mesmo efeito da operação *reserve* do pseudo-código.

- Em Pascal, a operação que libera memória é implementada por uma função, na forma:

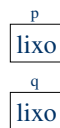
```
dispose(p);
```

que tem o mesmo efeito da operação *libere* do pseudo-código.

Alocação dinâmica com ponteiros em Pascal

Exemplo

```
var
  p,q:^integer;
```

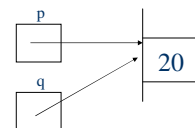


```
new(p);
p^ := 10;
q := p;
q^ := p^ + 10;
writeln(p^);
dispose(p);
q := nil;
```

Alocação dinâmica com ponteiros em Pascal

Exemplo

```
var
  p,q:^integer;
```



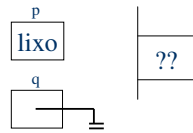
```
new(p);
p^ := 10;
q := p;
q^ := p^ + 10;
writeln("Resultado = ",p^);
dispose(p);
q := nil;
```

Resultado = 20

Alocação dinâmica com ponteiros em Pascal

Exemplo

```
var
  p,q:^integer;
```



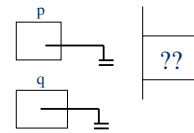
```
new(p);
p^ := 10;
q := p;
q^ := p^ + 10;
writeln("Resultado = ",p^);
dispose(p);
q := nil;
```

Resultado = 20

Alocação dinâmica com ponteiros em Pascal

Exemplo

```
var
  p,q:^integer;
```



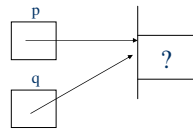
```
new(p);
p^ := 10;
q := p;
q^ := p^ + 10;
writeln("Resultado = ",p^);
dispose(p);
q := nil;
p:= nil; {por garantia}
```

Resultado = 20

Alocação dinâmica com ponteiros em Pascal

O que acontece com a adição da linha abaixo?

```
var
  p,q:^integer;
```



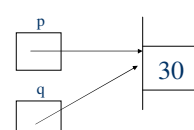
```
new(p);
p^ := 10;
q := p;
p^ := p^ + 10;
q^ := p^ + 10;
writeln("Resultado = ",p^);
```

Resultado = ?

Alocação dinâmica com ponteiros em Pascal

O que acontece com a adição da linha abaixo?

```
var
  p,q:^integer;
```



```
new(p);
p^ := 10;
q := p;
p^ := p^ + 10;
q^ := p^ + 10;
writeln("Resultado = ",p^);
```

Resultado = 30

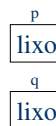
Alocação dinâmica com ponteiros em Pascal

Exemplo 2

```
type
  pont_rec = ^rec; {definição separada do
                    tipo ponteiro}

  rec = record
    dado: real;
    pont: pont_rec;
  end

var
  p,q : pont_rec;
```

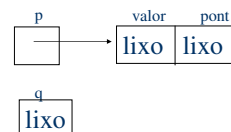


Alocação dinâmica com ponteiros em Pascal

Exemplo 2

```
var
  p,q :pont_rec;

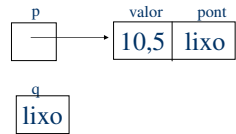
new(p);
p^.valor := 10.5;
new(q);
p^.pont := q;
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2

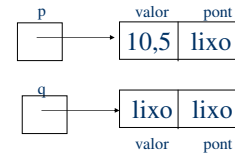
```
var
    p,q: pont_rec;
new(p);
p^.valor := 10.5;
new(q);
p^.pont := q;
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2

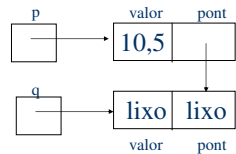
```
var
    p,q: pont_rec;
new(p);
p^.valor := 10.5;
new(q);
p^.pont := q;
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2

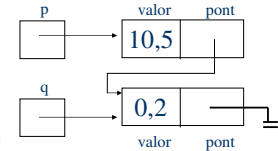
```
var
    p,q: pont_rec;
new(p);
p^.valor := 10.5;
new(q);
p^.pont := q;
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2

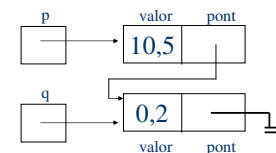
```
var
    p,q: pont_rec;
new(p);
p^.valor := 10.5;
new(q);
p^.pont := q;
q^.valor := 0.2;
q^.pont := Nil;
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2

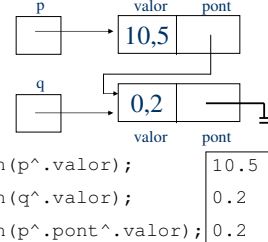
```
writeln(p^.valor);
writeln(q^.valor);
writeln(p^.pont^.valor);
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2 – Encadeando Registros em Pascal

```
writeln(p^.valor);
writeln(q^.valor);
writeln(p^.pont^.valor);
```

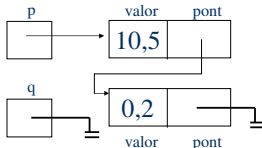


Alocação dinâmica com ponteiros em Pascal

Exemplo 2 – anulando q ainda se consegue acesso ao segundo dado armazenado

```
var
  p, q: pont_rec;

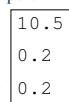
new(p);
p^.valor := 10.5;
new(q);
p^.pont := q;
q^.valor := 0.2;
q^.pont := Nil;
q := Nil;
```



Alocação dinâmica com ponteiros em Pascal

Exemplo 2

```
writeln(p^.valor);
writeln(q^.valor);
writeln(p^.pont^.valor);
```



Alocação dinâmica com ponteiros - questões

- No exemplo anterior:
 - O que acontece se, ao final, for executado o comando dispose (p)?
 - Como faríamos para destruir todos os dados voltando o espaço ocupado para a Heap e mantendo no final os ponteiros 'seguros'?

Alocação dinâmica com ponteiros – questão

- Conforme o que foi visto até agora, imagine o seguinte problema:
 - Como re-implementar uma aplicação que anteriormente utilizou vetor para realizar a mesma tarefa sem precisar definir antecipadamente uma quantidade máxima de elementos?

Lista Encadeada

```
type
  nome = string[20];
  apont_no = ^ no;
  no = record
    dado: nome;
    prox: apont_no;
  end;
  lista = apont_no;

Function vazia(L: lista): boolean;
{e: L: lista: ponteiro para o primeiro elemento da lista}
begin
  vazia := (L = NULL);
end;
```

Lista Encadeada

```
Function tamanho(L: lista): integer;
{e: L: lista: ponteiro para o primeiro elemento da lista}
{r: número de elementos da lista}
var
  cont: inteiro;
  pont: apont_no; {apontador auxiliar}
begin
  cont := 0;
  pont := L;
  while pont <> NULL do
  begin
    cont := cont + 1;
    pont := pont^.prox;
  end;
  tamanho := cont;
end;
```

Lista Encadeada

```
Procedure insere(L:lista;novo_nome:nome);
{e: novo_nome : nome}
{e/s: L:lista a lista sai alterada, o próprio ponteiro da lista
pode sair também}
var
  aux, pont, novo_p : apont_elemento;

begin
  if vazia(L) then {Tratamento de lista vazia}
  begin
    new(novo_p);
    novo_p.dado := novo_nome;
    novo_p.prox := NULL;
    L := novo_p;
  end
  else
  begin
```

Lista Encadeada

```
    aux := NULL;
    pont := L;
    while pont.dado < novo_nome and pont.prox <> NULL do
      begin
        aux := pont;
        pont := pont.prox;
      end
    if pont.dado <> novo_nome then
      begin
        new(novo_p);
        novo_p.dado := novo_nome;
        if aux <= NULL then
          begin
            novo_p.prox := aux.prox
            aux.prox := novo_p
          end
        else
          begin
            novo_p.prox := L;
            L := novo_p;
          end
        end
      end
    end
  end
```

Lista Encadeada

```
Procedure elimina(L:lista,nome,el:nome)
{e: nome_el : nome}
{e/s: L:lista a lista sai alterada, o próprio ponteiro da lista
pode sair também}
var
  aux, pont, novo_p : apont_elemento;

begin
  if not vazia(L) then {Teste de lista vazia}
  {Encontre o elemento a ser eliminado }
  aux := NULL;
  pont := L;
  while pont.dado < novo_nome and pont.prox <> NULL do
    begin
      aux := pont;
      pont := pont.prox;
    end;
  {Elimine o elemento }
  if pont.dado = novo_nome then {o elemento existe}
  begin
    if aux <= NULL then {o elemento não é o primeiro}
      aux.prox := pont.prox
    else
      L := pont.prox;
      {devolva o espaço ocupado pelo elemento à heap}
      release(pont);
    end;
  end
end
```

Exercício Sugerido

- Implemente o seu algoritmo para a busca e eliminação por posição, o invés de conteúdo