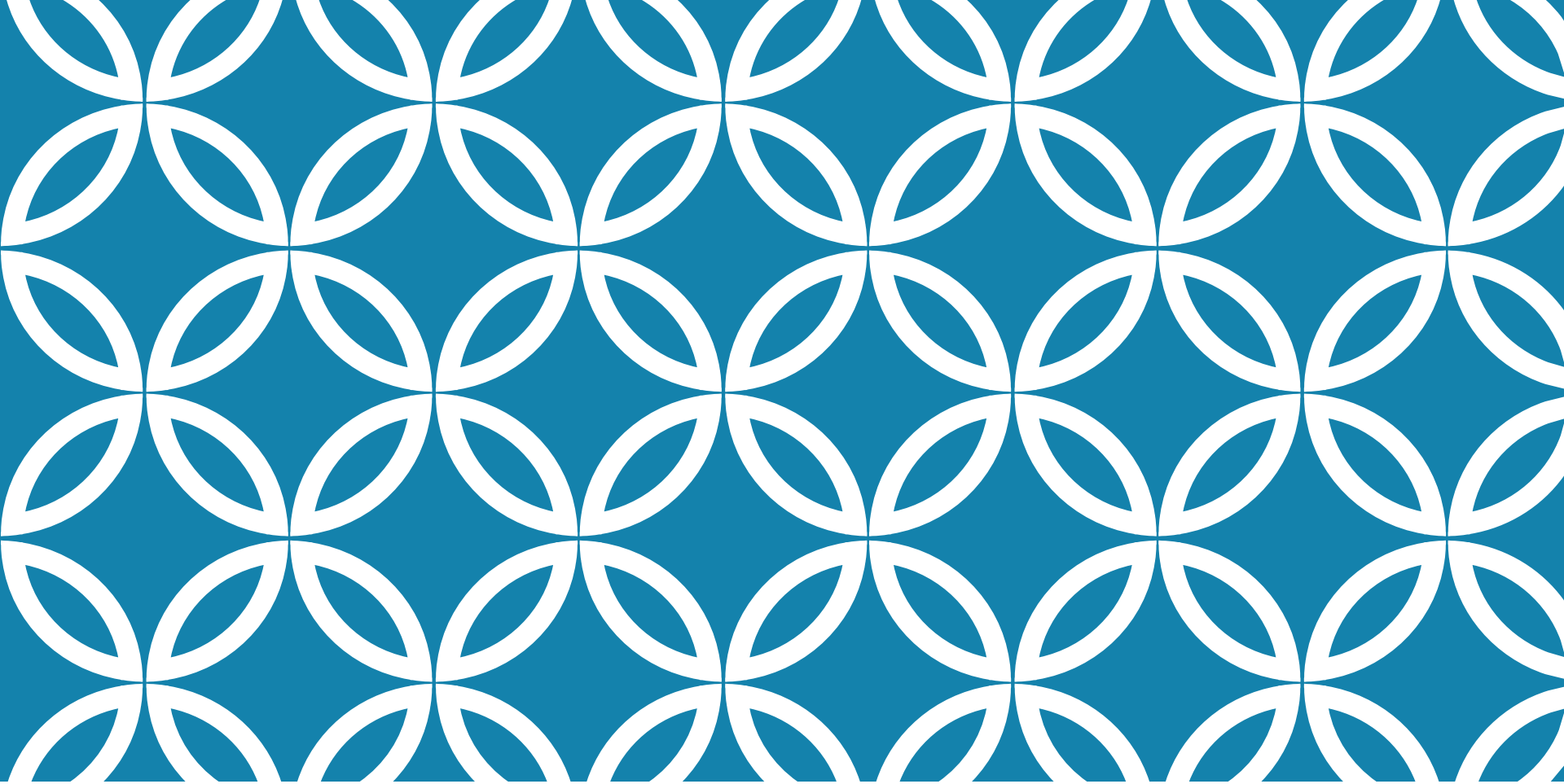

IMAGENS

Prof. André Vignatti

EXEMPLO DE ARQUIVO PGM

```
P2
11 10
40
40 5  5  5  5  5  5  5  5  40 0
5  20 20 5  5  5  5  5  5  5  5
5  5  20 5  5  5  0  0  0  0  0
5  5  20 20 5  5  20 20 0  0  5
5  5  5  5  5  5  0  20 0  0  0
5  5  5  5  5  5  0  20 20 0  5
5  5  5  5  11 11 11 0  0  0  0
5  5  5  5  20 20 11 5  5  5  5
5  5  5  5  11 20 11 5  5  5  0
40 5  5  5  11 20 20 5  5  40 5
```

MAIOR VALOR

```
function maior_valor (var O: imagem; l, c : integer) : integer;
var
    i, j, m: integer;
begin
    m:= O[1,1];
    for i:= 1 to l do
        for j:= 1 to c do
            if O[i,j] > m then
                m := O[i,j] ;
    maior_valor:= m;
end;
```

ZOOM OUT

```
procedure zoom_pgm (var O: imagem; lO,cO : integer; var D: imagem; var lD, cD, maxD: integer) ;
var
    i, j : integer;
begin
    lD:= lO div 2;
    cD:= cO div 2;
    for i:= 1 to lD do
        for j:= 1 to cD do
            D[i,j] := media_4_vizinhos (O, i, j);
    maxD:= maior_valor(D, lD, cD);
end;
```

```
function media_4_vizinhos (var O : imagem; i, j : integer) : integer;
var
    x, y: integer;
begin
    x := 2*i - 1;
    y := 2*j - 1;
    media_4_vizinhos := (O[x,y] + O[x+1,y] + O[x,y+1] + O[x+1,y+1]) div 4;
end;
```

DETECÇÃO DE BORDAS

```
procedure detectar_bordas_pgm (var O, D: imagem; l, c : integer; var max: integer);
var
    i , j , grad: integer;
begin
(* bordas recebem zero *)
    for i := 1 to l do begin
        D[i,1] := 0;
        D[i,c] := 0;
    end;

    for i := 1 to c do begin
        D[1,i] := 0;
        D[l,i] := 0;
    end;

    for i := 2 to l-1 do
        for j := 2 to c-1 do begin
            grad:= abs (O[i,j]*4 - (O[i-1,j] + O[i+1,j] + O[i,j-1] + O[i,j+1])) ;
            if grad > limiar then
                D[i,j] := 255
            else
                D[i,j] := 0;
        end;
    max:= 255;
end;
```

TIPO CHAR

char: tipo para armazenar caracteres

```
var ch: char;

begin
  ch := 'A';
end.
```

Geralmente lidamos com vários caracteres. Usando vetor:

```
type
  string4 = array[0..3] of char;
var
  s: string4;
begin
  s[0] := 'a';
  s[1] := 'b';
  s[2] := 'c';
  s[3] := 'd';
end;
```

TIPO STRING

string: tipo para sequência de caracteres

```
var
  str1, str2: string[10];
begin
str1 := 'abc';
str2 := '123';
end.
```

Usar **string** facilita nossa vida:

```
var
  s, str1, str2: string[10];
  c: char;
  n: integer;
begin
str1 := 'abc';
str2 := '123';
s := str1 + str2;  // concatenacao
c := s[1];         // usa como indice de vetor
n := length( s );  // tamanho da string s
end.
```

TIPOS COMPOSTOS (HETEROGÊNEOS)

Até agora em CI055, vimos tipos homogêneos

- Ou seja, guardam valores de um único tipo
- Mesmo os vetores vistos são homogêneos

Porque seria interessante ter dados heterogêneos?

- Criação de tipos complexos
- Abstração maior à medida que o programa cresce

REGISTROS – DEFINIÇÃO DE TIPO E ALOCAÇÃO DE VARIÁVEIS

Em Pascal: **record**

Toda linguagem de programação permite criar tipos heterogêneos

```
type cliente = record
        nome: string[50];
        fone : longint ;
        endereco: string[100];
        sexo : char;
        idade : integer;
        rg : longint;
        cpf : string[20];
end;

var r : cliente ;
```

REGISTROS – LENDO VALORES

Atribuição direta:

```
r.nome:= 'Fulano de Tal';
r.fone:= 32145678;
r.endereco:= 'Rua dos bobos, no 0';
r.sexo:= 'M';
r.idade:= 75;
r.rg:= 92346539;
r.cpf:= '111.222.333-44';
```

Lendo da entrada padrão:

```
read (r.nome);
read (r.fone);
read (r.endereco);
read (r.sexo);
read (r.idade);
read (r.rg);
read (r.cpf);
```