ATIVIDADE EXTRA-CLASSE

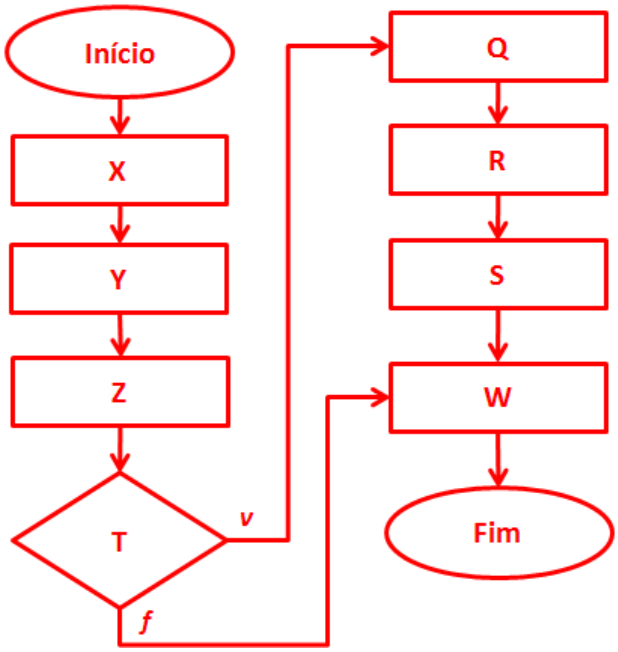
2 – Classes de Programas
GABARITO

1- Dados os programas iterativos a seguir; transforme-os para Programas Monolíticos.

a-)

```
P1A = (
X; Y; Z; se T (Q; R; S) senão ✓; W)
```

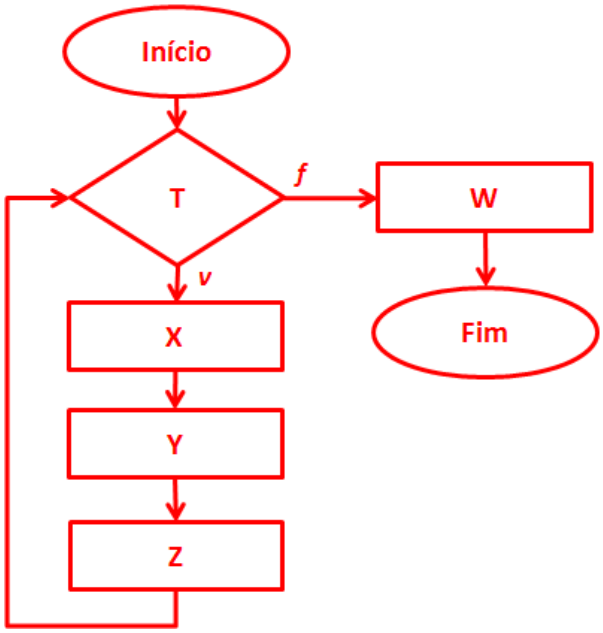
1: faça X; vá_para 2
2: faça Y; vá_para 3
3: faça Z; vá_para 4
4: se T então vá_para 5 senão vá_para 8
5: faça Q; vá_para 6
6: faça R; vá_para 7
7: faça S; vá_para 8
8: faça W; vá_para 9



b-)

```
P1B = (
      enquanto T faça (X; Y; Z); W)
```

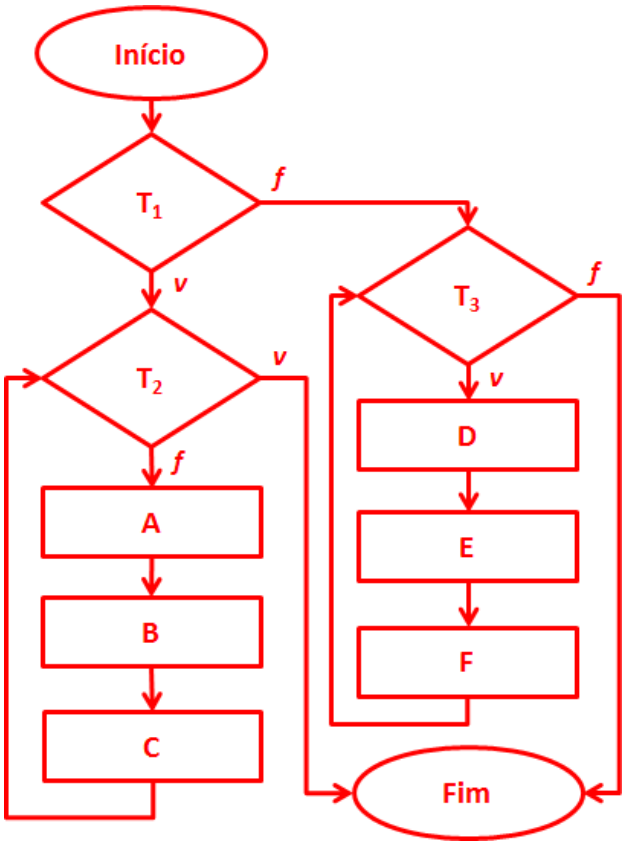
1: se T então vá_para 2 senão vá_para 5
2: faça X; vá_para 3
3: faça Y; vá_para 4
4: faça Z; vá_para 1
5: faça W; vá_para 6



c-)

```
P1c = (
      se T1 (
         até T2 faça (A; B; C))
      senão (
         enquanto T3 faça (D; E; F))
)
```

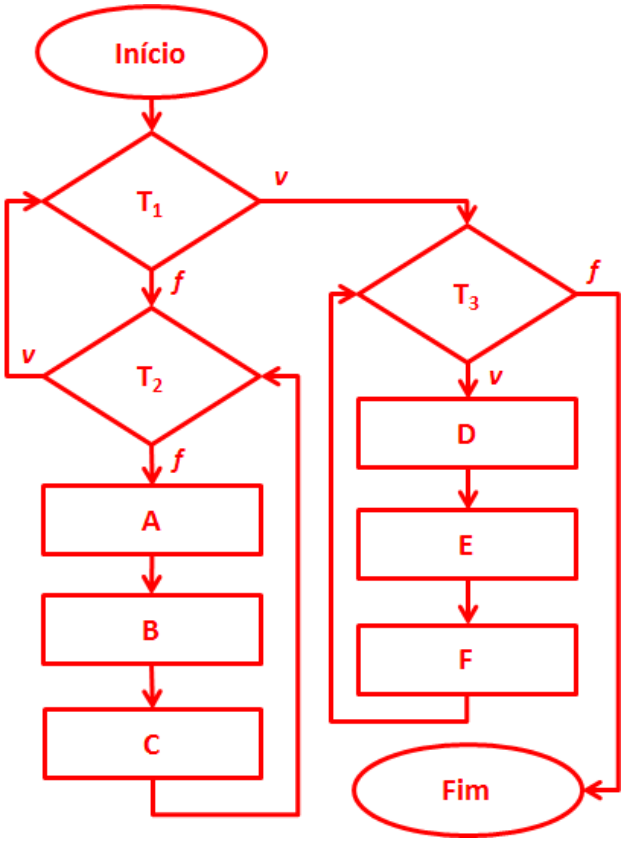
1: se T_1 então vá_para 2 senão vá_para 6
2: se T_2 então vá_para 10 senão vá_para 3
3: faça A; vá_para 4
4: faça B; vá_para 5
5: faça C; vá_para 2
6: se T_3 então vá_para 7 senão vá_para 10
7: faça D; vá_para 8
8: faça E; vá_para 9
9: faça F; vá_para 6



d-)

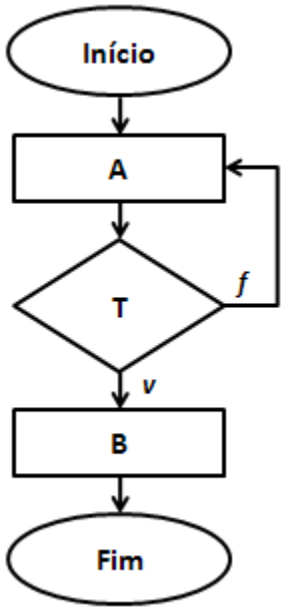
```
P1D = (
      até T1 faça (
         até T2 faça (A; B; C));
      enquanto T3 faça (D; E; F)
)
```

1: se T_1 então vá_para 6 senão vá_para 2
2: se T_2 então vá_para 1 senão vá_para 3
3: faça A; vá_para 4
4: faça B; vá_para 5
5: faça C; vá_para 2
6: se T_3 então vá_para 7 senão vá_para 10
7: faça D; vá_para 8
8: faça E; vá_para 9
9: faça F; vá_para 6



2-) Dados os fluxogramas de Programas Monolíticos a seguir; transforme-os para Programas Recursivos.

a-)



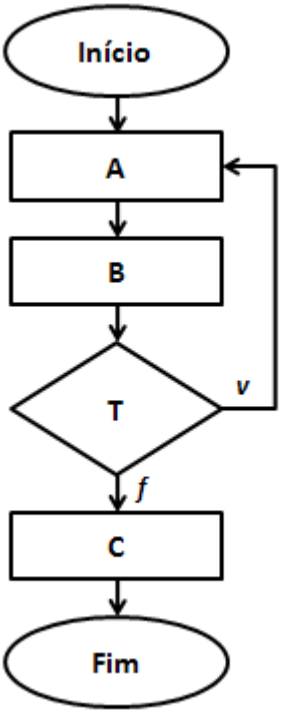
1: faça A; vá_para 2
2: se T então vá_para 3 senão vá_para 1
3: faça B; vá_para 4

P é R_1 onde
R_1 def (A; R_2)
R_2 def (se T R_3 senão R_1)
R_3 def (B; R_4)
R_4 def (✓)

De forma resumida:

P é R onde
R def (A; se T (B; ✓) senão R)

b-)



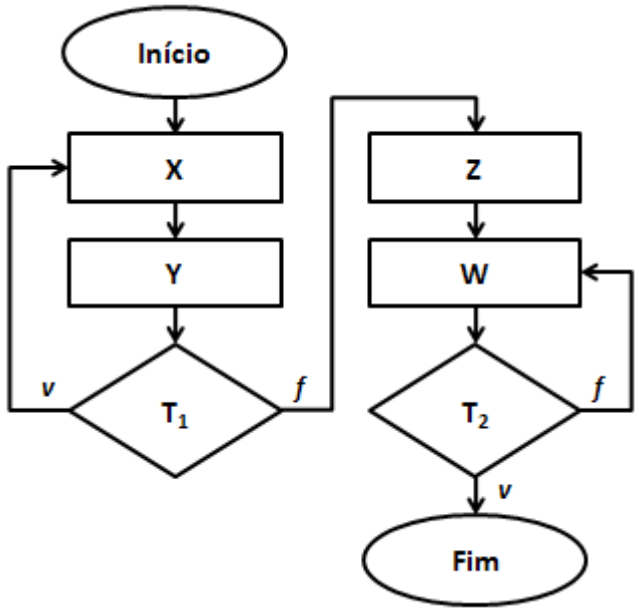
1: faça A; vá_para 2
2: faça B; vá_para 3
3: se T então vá_para 1 senão vá_para 4
4: faça C; vá_para 5

P é R_1 onde
R_1 def (A; R_2)
R_2 def (B; R_3)
R_3 def (se T R_1 senão R_4)
R_4 def (C; R_5)
R_5 def (✓)

De forma resumida:

P é R onde
R def (A; B; se T R senão (C; ✓))

c-)



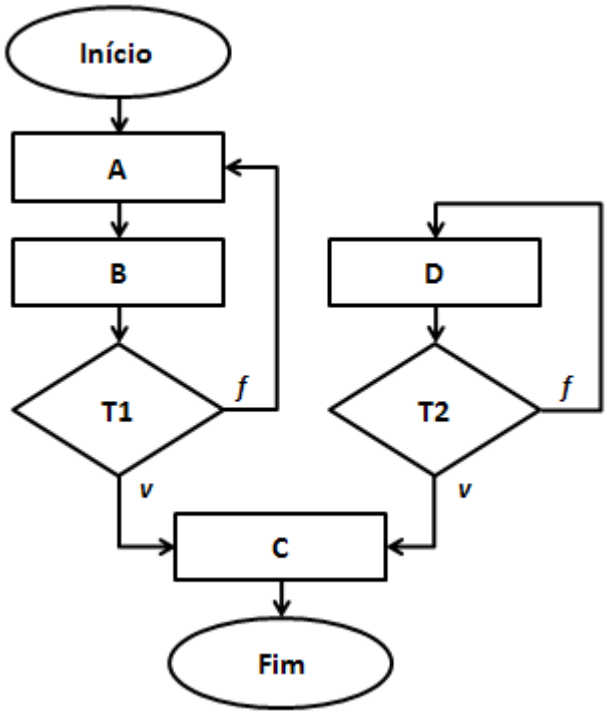
1: faça X; vá_para 2
2: faça Y; vá_para 3
3: se T_1 então vá_para 1 senão vá_para 4
4: faça Z; vá_para 5
5: faça W; vá_para 6
6: se T_2 então vá_para 7 senão vá_para 5

P é R_1 onde
R_1 def (X; R_2)
R_2 def (Y; R_3)
R_3 def (se T_1 R_1 senão R_4)
R_4 def (Z; R_5)
R_5 def (W; R_6)
R_6 def (se T_2 R_7 senão R_5)
R_7 def (✓)

De forma resumida:

P é R onde
R def (X; Y; se T_1 R senão (Z; S))
S def (W; se T_2 ✓ senão S)

d-)



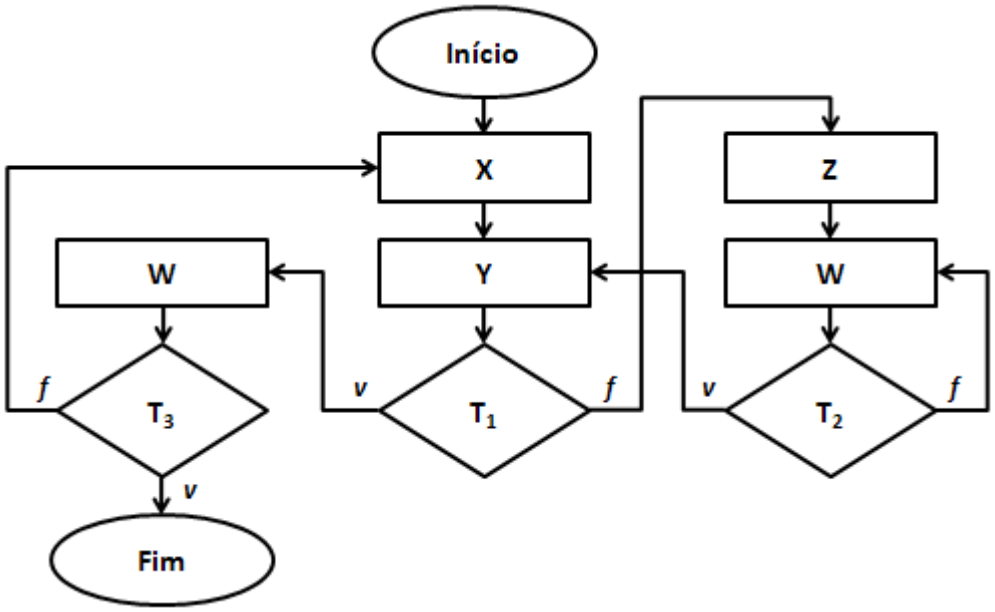
1: faça A; vá_para 2
2: faça B; vá_para 3
3: se T1 então vá_para 4 senão vá_para 1
4: faça C; vá_para 7
5: faça D; vá_para 6
6: se T2 então vá_para 4 senão vá_para 5

P é R_1 onde
R_1 def (A; R_2)
R_2 def (B; R_3)
R_3 def (se T_1 R_4 senão R_1)
R_4 def (C; R_7)
R_5 def (D; R_6)
R_6 def (se T_2 R_4 senão R_5)
R_7 def (✓)

De forma resumida:

P é R onde
R def (A; B; se T_1 (C; ✓) senão R)
S def (D; se T_2 (C; ✓) senão S)

e-)



1: faça X; vá_para 2
2: faça Y; vá_para 3
3: se T_1 então vá_para 7 senão vá_para 4
4: faça Z; vá_para 5
5: faça W; vá_para 6
6: se T_2 então vá_para 2 senão vá_para 5
7: faça W; vá_para 8
8: se T_3 então vá_para 9 senão vá_para 1

P é R_1 onde
R_1 def (X; R_2)
R_2 def (Y; R_3)
R_3 def (se T_1 R_7 senão R_4)
R_4 def (Z; R_5)
R_5 def (W; R_6)
R_6 def (se T_2 R_2 senão R_5)
R_7 def (W; R_8)
R_8 def (se T_3 R_9 senão R_1)
R_9 def (✓)

De forma resumida:

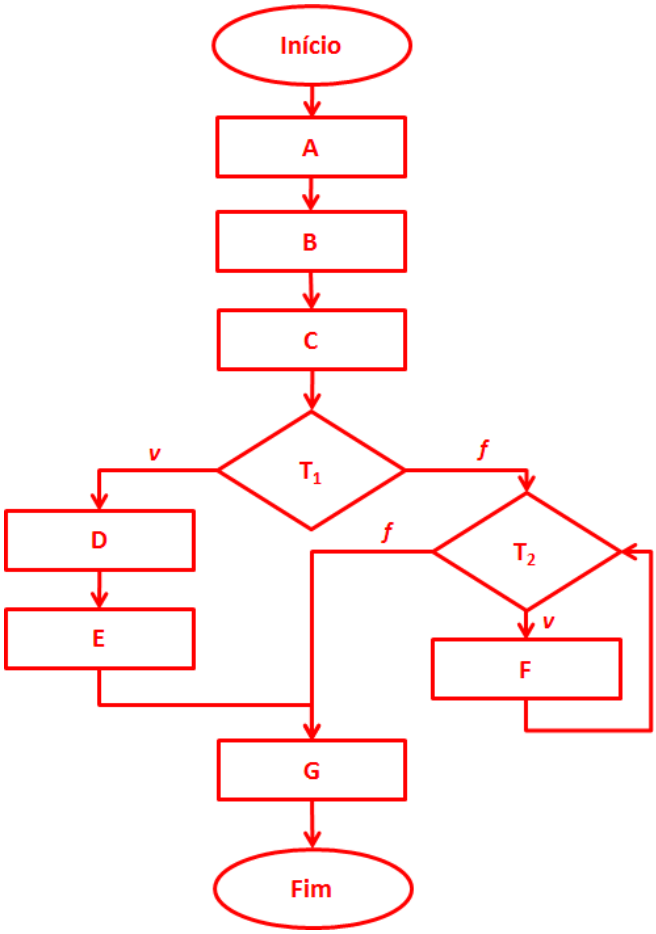
P é R onde
R def (X; (Y; se T_1 (W; se T_3 ✓ senão R) senão (Z; S)))
S def (W; se T_2 (Y; se T_1 (W; se T_3 ✓ senão R) senão (Z; S)) senão S)

3-) Dado o Programa Iterativo a seguir, converta-o para um Programa Recursivo.

```
P_ITER = (
    A; B; C; se T1 (D; E) senão (
        enquanto T2 (F)
    ); G
)
```

Iterativo → Monolítico

1: faça A; vá_para 2
2: faça B; vá_para 3
3: faça C; vá_para 4
4: se T_1 então vá_para 5 senão vá_para 7
5: faça D; vá_para 6
6: faça E; vá_para 9
7: se T_2 então vá_para 8 senão vá_para 9
8: faça F; vá_para 7
9: faça G; vá_para 10



Monolítico → Recursivo

P é R_1 onde
R_1 def (A; R_2)
R_2 def (B; R_3)
R_3 def (C; R_4)
R_4 def (se T_1 R_5 senão R_7)
R_5 def (D; R_6)
R_6 def (E; R_9)
R_7 def (se T_2 R_8 senão R_9)
R_8 def (F; R_7)
R_9 def (G; R_{10})
R_{10} def (✓)

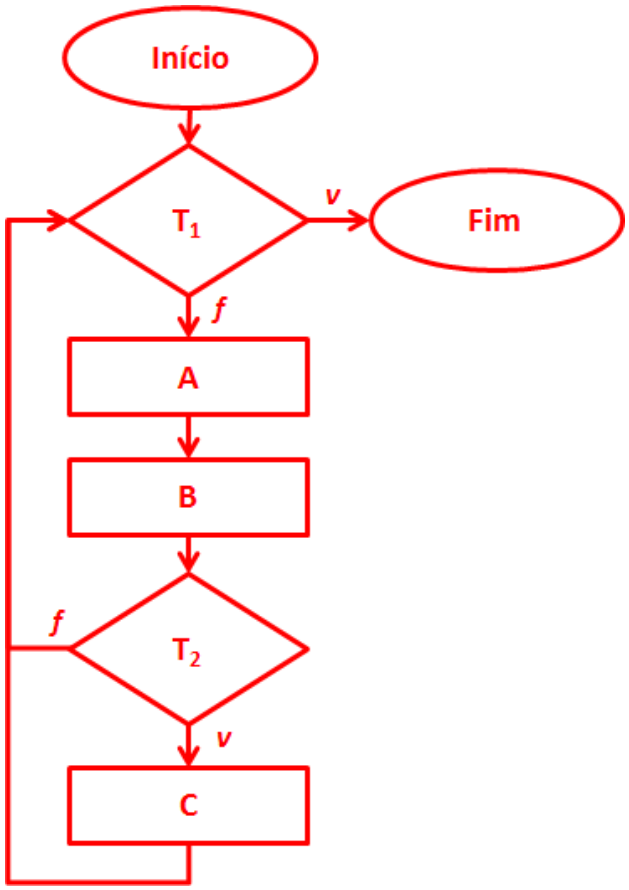
De forma resumida:

P é R onde
R def (A; B; C; se T_1 (D; E; G; ✓) senão S)
S def (se T_2 (F; S) senão (G; ✓))

4-) Dado o trecho de código em Pascal a seguir, converta-o para um Programa Recursivo.

```
var
sexo: string;
begin
   repeat
      write('Sexo (m/f):');
      readln(sexo);
      if (sexo <> 'm') and (sexo <> 'f') then begin
         writeln('Sexo deve ser m ou f');
      end;
   until (sexo = 'm') or (sexo = 'f');
end.
```

$P_{ITERATIVO}$ = (
até T_1 faça (
A; B; se T_2 C senão ✓)
)



Iterativo → Monolítico

1: se T_1 então vá_para 2 senão vá_para 6
2: faça A; vá_para 3
3: faça B; vá_para 4
4: se T_2 então vá_para 5 senão vá_para 1
5: faça C; vá_para 1

Monolítico → Recursivo

P é R_1 one
R_1 def (se T_1 R_2 senão R_6)
R_2 def (A; R_3)
R_3 def (B; R_4)
R_4 def (se T_2 R_5 senão R_1)
R_5 def (C; R_1)
R_6 def (✓)

De forma resumida:

P é R one
R def (se T_1 (A; B; se T_2 (C; R) senão R) senão ✓)