

Programación 1. Primer Parcial 2025

Instituto de Computación

Casilla de control: 1

Ejercicio 1. Considere x de tipo integer y b de tipo boolean. Indique cuál de las siguientes asignaciones **puede dar error** al ejecutarse:

- A) $b := (x > 0) \text{ and } (17 \text{ div } x < 2)$
- B) $b := (x <> 0) \text{ and } (17 \text{ div } x > 2)$
- C) $b := (x = 0) \text{ or } (17 \text{ div } x > 2)$
- D) $b := (17 \bmod (\text{sqr}(x) - 1) > 2) \text{ or } (x = 1)$
- E) $b := (x \bmod 2 <> 0) \text{ and } (17 \text{ div } x > 0)$

Ejercicio 2. Considere la siguiente declaración de variables:

```
var a,b,c : integer;
```

y los siguientes encabezados de subprogramas:

```
procedure proc (x : integer; var y, z : integer);
function fun (x, y : integer) : integer;
```

Indique cuál de las siguientes instrucciones **es correcta**:

- A) `proc (2, b, fun (a, 4))`
- B) `a := fun (b, proc (4, b, c))`
- C) `proc (fun (a + b, c), a, b)`
- D) `c := proc (a, b, c)`
- E) `fun (a, 9)`

Ejercicio 3. Considere el siguiente fragmento de código, donde todas las variables son enteras:

```
z := -1;
cont := 0;
for i := 1 downto -2 do
begin
  k := i * z;
  while k < 0 do
  begin
    cont := cont + 1;
    k := k + 1
  end;
  writeln(k);
  z := z * -1
end;
```

Indique la afirmación **correcta**.

- A) No se despliega más de una vez el mismo valor.
- B) El valor de k al finalizar el `for` es indeterminado.
- C) Al finalizar el `for`, los valores de `cont` y `k` coinciden.
- D) Entre todas las evaluaciones de $k < 0$, hay más resultados `true` que `false`.
- E) El valor de `cont` al finalizar la ejecución es impar.

Ejercicio 4. Considere el siguiente esquema de programa:

```
<inicio>
while <cond1> do
  repeat
    <cuero1>
  until <cond2>;
while <cond2> do
  <cuero2>
```

Indique la afirmación que **no es correcta**:

- A) Si el primer `while` termina luego de iterar al menos una vez, `<cuero2>` se va a ejecutar por lo menos una vez
- B) Puede suceder que `<cuero1>` no se ejecute
- C) Puede suceder que `<cuero2>` no se ejecute
- D) Si en `<cuero1>` no se modifican las variables de `<cond1>` y `<cond2>`, `<cuero2>` nunca se va a ejecutar
- E) Si luego de ejecutado `<inicio>` la condición `<cond1>` es `true`, `<cuero1>` se va a ejecutar por lo menos una vez

Ejercicio 5. Considere dos expresiones booleanas $e1$ y $e2$, dos instrucciones `instr1` e `instr2`, y el siguiente fragmento de código:

```
if e1 or e2 then
  instr1
else
  instr2
```

¿Cuál de los siguientes fragmentos es equivalente?

- A)

```
if e1 then
  instr1
else
  if e2 then
    instr2
  else
    instr1
```
- B)

```
if e1 then
  instr1
else
  if e2 then
    instr1
  else
    instr2
```
- C)

```
if e1 then
  instr1;
if not e2 then
  instr2
```
- D)

```
if e1 then
  instr1;
if e2 then
  instr2
```
- E)

```
if e1 then
  if e2 then
    instr1
  else
    instr2
```

Ejercicio 6. Considere el siguiente fragmento de código, donde todas las variables son enteras:

```
for i := n downto 0 do
  if(i mod 2 = 0) then
    writeln(i+1)
  else
    writeln(i);
```

Indique la afirmación **correcta**.

- A) Si $n = 0$, el programa despliega 0.
- B) Si $n = 0$, el programa no despliega nada.
- C) Si $n > 0$ y n es par, el programa despliega los números impares desde $n + 1$ hasta 1, imprimiendo cada uno de ellos dos veces.
- D) Si $n > 0$ y n es impar, el programa despliega los números impares desde n hasta 1, imprimiendo cada uno de ellos dos veces.
- E) Si $n < 0$, ocurre un error en tiempo de ejecución.

Ejercicio 7. Considere el siguiente programa:

```
program p;
var
  x,y : real;
  z : integer;
begin
  x := 3.6;
  y := 2.3;
  z := [expresión]
end.
```

Indique cuál es la [expresión] que asigna en la variable z un valor **distinto** de 6.

- A) $\text{trunc}(\text{round}(x) + y)$
- B) $\text{trunc}(y) + \text{round}(x + y) - \text{round}(y)$
- C) $\text{round}(x) + \text{trunc}(x - y)$
- D) $\text{trunc}(\text{trunc}(x) + x)$
- E) $\text{round}(y) + \text{round}(x)$

Ejercicio 8. Considere x de tipo `integer`. Indique cuál de las siguientes asignaciones **es correcta**:

- A) $x := (2 < x) \text{ and } (x < 10)$
- B) $x := \text{write}(10)$
- C) $x := \text{round}(x \text{ div } 10) + \text{trunc}(x / 10)$
- D) $x := (x / 10) * 10 + x \bmod 10$
- E) $x := \text{if } x < 10 \text{ then } x \text{ else } 10$

Ejercicio 9. Considere el siguiente fragmento de código, donde todas las variables son enteras:

```
read(a, b);
for j := 2 to 3 do
begin
  a := a + j;
  write(b + a + j);
  for i := 3 downto 2 do
  begin
    b := b - i;
    write(a - b + j)
  end;
end;
```

Considere que la entrada está dada por los enteros 2 y 5. Indique la afirmación **correcta**.

- A) La salida está dada por los números 11 4 6 10 13 15.
- B) La salida está dada por los números 11 10 12 10 19 21.
- C) La salida está dada por los números 11 3 6 10 12 15.
- D) El fragmento presenta errores en tiempo de ejecución.
- E) El fragmento presenta errores en tiempo de compilación.

Ejercicio 10. Considere el siguiente programa:

```
program p;
var x : integer;
begin
  x := 5;
  while x mod 3 <> 0 do
    case x mod 3 of
      0: x := x mod 3;
      1: x := x mod 3 + 1;
      2: x := x mod 3 + 2;
    end
  end.
end.
```

Determine cuántas veces se ejecuta la instrucción `case`:

- A) 1
- B) 2
- C) 3
- D) ninguna
- E) infinitas

Ejercicio 11. Considere el siguiente fragmento de código, donde todas las variables son enteras:

```
y := 3;
for x := 0 to 2 do
begin
  repeat
    write('o');
    y := y - 1
  until y < x;
  y := x;
  write('$')
end;
```

Indique la afirmación **correcta**.

- A) Imprime infinitos o (y no termina)
- B) Imprime oooo\$o\$o\$
- C) Imprime ooo\$o\$o\$
- D) Imprime o\$o\$o\$
- E) Imprime oooo\$\$\$

Ejercicio 12. Considere el siguiente fragmento de código donde todas las variables son enteras:

```
x := 1;
y := 0;
while x <= 10 do
begin
  y := y + 1;
  while y mod 3 <> 0 do
  begin
    x := x + 1;
    y := y + 1
  end
end;
write(x, '-', y)
```

Indique la afirmación **correcta**.

- A) Imprime 11-15
- B) Imprime 10-15
- C) Imprime 11-9
- D) Imprime 10-9
- E) No imprime nada porque se produce una iteración infinita

Ejercicio 13. Dado el siguiente fragmento de código, donde *n* y *c* son variables de tipo *integer* y *char* respectivamente:

```
while a (8.5, n) do  
  e (c = '0', n + 1)
```

Indique cuál de los siguientes pares de encabezados es **correcto**:

- A)

function a (r : real; i : integer) : boolean; procedure e (d : char; i : integer);

- B)

function a (r : real; i : integer) : boolean; procedure e (b : boolean; var i : integer);
--
- C)

function a (r,i : real) : boolean; procedure e (b : boolean; r : real);
--
- D)

function a (var r : real; i : integer) : boolean; procedure e (b : boolean; i : integer);
--
- E)

function a (r,i : real) : boolean; procedure e (d : char; r : real);

Ejercicio 14. Considere el siguiente fragmento de código, donde todas las variables son enteras:

```
if (a = b) and (b <> c) then write('1')  
else if a = b then write('2')  
    else write('3');  
if (b <> c) then write('4')  
else write ('5')
```

Indique cuál salida **no** se dará **para ningún** valor de *a*, *b* y *c*:

- A) 14
- B) 24
- C) 34
- D) 35
- E) 25

Ejercicio de Resolución

- El puntaje máximo por este ejercicio es **12 puntos** (no resta puntos).
- En este ejercicio se evaluará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera se tendrá en cuenta entre otros conceptos: indentación apropiada, correcta utilización de las estructuras de control, código elegante y legible, eficiencia de los algoritmos, pasaje correcto de parámetros, etcétera.

En el laboratorio se había definido el tipo `Natural`. Con ese tipo se debía programar el procedimiento `siguienteDigito` con el siguiente encabezado:

```
procedure siguienteDigito(var num: Natural; var digito: integer);
```

Parte A. Implemente el procedimiento `siguienteDigito`.

Parte B. Implemente la función:

```
function esCantidadDe (cant : integer; dig : integer; num : Natural) : boolean;
```

que recibe una cantidad `cant` de ocurrencias de un dígito, el dígito `dig` y un número natural estrictamente positivo `num`. La función devuelve `true` si el número `num` tiene exactamente la cantidad `cant` de ocurrencias de `dig`, y `false` en caso contrario.

Ejemplos de ejecución de la función `esCantidadDe`:

- `esCantidadDe(2,8,808819)`, retorna `false`
- `esCantidadDe(3,8,808819)`, retorna `true`
- `esCantidadDe(1,9,808819)`, retorna `true`