Práctica 4 - no entendí nada pero finjamos que sí

domingo, 7 de abril de 2024 16:29

Ejercicio 1: a) Tome una de las variables de la línea 3 del siguiente código e indique y defina cuales son sus atributos:

```
1. Procedure Practica4();
2. var
3. a,i:integer
4. p:puntero
5. Begin
6. a:=0;
7. new(p);
8. p:= ^i
9. for i:=1 to 9 do
10.a:=a+i;
11.end;
12.....
13.p:= ^a;
14.....
15.dispose(p);
16.end;
```

nombre: a
alcance: 3-16
L-valor: automática
R-valor: indef-0
Tiempo de vida:1-16

b) Compare los atributos de la variable del punto a) con los atributos de la variable de la línea 4. Que dato contiene esta variable?,

- Estática (lenguaje C)
- Automática (variables comunes)
- Dinámica (punteros)
- Semidinámica (arreglos semidinámicos ADA)

Nombre: p
alcance: 7-15
L-valor: Dinámica
R-valor: l-valor de i, l-valor de a
Tiempo de vida: 7-15???

a. Indique cuales son las diferentes formas de inicializar una variable en el momento de la declaración de la misma.

- Ignorar el problema: la inicializo con lo que haya en memoria
- Estrategia de inicialización:
  - Inicialización por defecto: Enteros se inicializan en 0, los caracteres en blanco, etc.
  - Inicialización en la declaración: C int i =0, j= 1

b. Analice en los lenguajes: Java, C, Phyton y Ruby las diferentes formas de inicialización de variables que poseen. Realice un cuadro comparativo de esta característica.

Java	- Declaración e inicialización en la misma línea: int num = 10; - Declaración e inicialización separadas: int num; num = 10; - Inicialización por defecto: int num; (inicializada a 0 automáticamente)
C	- Declaración e inicialización en la misma línea: int num = 10; - Declaración e inicialización separadas: int num; num = 10; - Las variables globales y estáticas se inicializan por defecto a 0 si no se les asigna un valor explícitamente. - Las variables locales no tienen un valor por defecto y su valor inicial es impredecible, a menos que se les asigne un valor explícitamente.
Python	- Declaración e inicialización en la misma línea: num = 10 - No es necesario declarar el tipo de variable antes de su inicialización
Ruby	- Declaración e inicialización en la misma línea: num = 10 - No es necesario declarar el tipo de variable antes de su inicialización

Ojo con variables declaradas tipo n:= int 6. vss n:= 6
Una de esas es en ejecucion y la otra en ompilacion, por lo que una tiene de n valor basura

Ejercicio 3: Explique los siguientes conceptos asociados al atributo l-valor de una:

a. Variable estática.

Una variable estática tiene un ámbito de visibilidad limitado a la función o archivo en el que se declara y su valor se mantiene constante durante toda la ejecución del programa. Es decir, es una variable que existe y se inicializa en tiempo de compilación y su valor persiste en memoria durante toda la ejecución del programa.

Ejemplo en C:

```c
#include <stdio.h>

void static_example() {
  static int num = 0;
  printf("El valor de num es: %d\n", num);
  num++;
}

int main() {
  for (int i = 0; i < 5; i++) {
    static_example();
  }
  return 0;
}
```

En este ejemplo, la variable num es una variable estática dentro de la función static_example().

La salida del programa sería:

```
El valor de num es: 0
El valor de num es: 1
El valor de num es: 2
El valor de num es: 3
El valor de num es: 4
```

b. Variable automática o semiestática.

Una variable automática o semiestática se declara dentro de un bloque y su ámbito de visibilidad se limita a ese bloque. Su valor se inicializa al entrar al bloque y se destruye al salir de él. Estas variables también se llaman locales, ya que solo son visibles dentro de la función en la que se declaran.

Ejemplo en Python:

```python
def automatic_example():
    num = 0
    print("El valor de num es: ", num)
    num += 1

for i in range(5):
    automatic_example()
```

En este ejemplo, la variable num es una variable automática dentro de la función automatic_example().

La salida del programa sería:

```
El valor de num es: 0
El valor de num es: 0
El valor de num es: 0
El valor de num es: 0
El valor de num es: 0
```

c. Variable dinámica.

Una variable dinámica es una variable que se crea y se destruye en tiempo de ejecución. En la mayoría de los lenguajes de programación, estas variables se crean utilizando funciones o métodos específicos como malloc() o new. El valor de una variable dinámica puede cambiar a lo largo de la ejecución del programa.

Ejemplo en Python:

```python
def dynamic_example():
    num = input("Ingrese un número: ")
    print("El valor ingresado es:", num)

for i in range(5):
    dynamic_example()
```

En este ejemplo, la variable num es una variable dinámica que se inicializa mediante la entrada de usuario en cada llamada a la función dynamic_example().

La salida del programa dependerá de lo que ingrese el usuario.

d. Variable semidinámica.

Una variable semidinámica es aquella cuyo tamaño puede cambiar durante la ejecución del programa, pero solo se puede ajustar su tamaño en ciertos puntos de control predefinidos. Un ejemplo de una variable semidinámica en ADA podría ser una matriz cuyo tamaño se puede ajustar en tiempo de ejecución, pero solo en un punto de control específico del programa.

Ejemplo en ADA:

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Ejemplo_Variable_Semidinamica is
   type Matriz is array (Positive range <>, Positive range <>) of Integer;
   -- Definición de una matriz de tipo semidinámico

   Filas, Columnas : Positive := 5;  -- Tamaño inicial de la matriz
   M : Matriz (1..Filas, 1..Columnas); -- Inicialización de la matriz

begin
   -- Realizar alguna operación con la matriz inicial

   Put_Line("El tamaño actual de la matriz es " & Positive'Image(Filas) & "x" &
   Positive'Image(Columnas));

   -- En algún punto de control en el programa, ajustar el tamaño de la matriz
   Filas := 8;
   Columnas := 8;
   M := M (1..Filas, 1..Columnas); -- Ajustar el tamaño de la matriz

   -- Realizar alguna operación con la matriz actualizada

   Put_Line("El tamaño actual de la matriz es " & Positive'Image(Filas) & "x"
& Positive'Image(Columnas));
end Ejemplo_Variable_Semidinamica;
```

Ejercicio 7: Sea el siguiente segmento de código escrito en Java, indique para los identificadores si son globales o locales.

```
Clase Persona {
        public long id
        public string nombreApellido
        public Domicilio domicilio
        private string dni;
        public string fechaNac;
        public static int cantTotalPersonas;

        //Se tienen los getter y setter de cada una
de las variables
        //Este método calcula la edad de la persona
a partir de la fecha de nacimiento
```

```
public int getEdad(){
                public int edad=0;
                public string fN =
this.getFechaNac();
                        ...
                        ...
          return edad;
        }
}
Clase Domicilio {
        public long     id;
        public static int nro
        public string calle
        public Localidad loc;

        //Se tienen los getter y setter de cada una
de las variables
}
```

Globales (variables estáticas de clase, aquellos que se declaran a nivel de la clase y pueden ser accedidos sin necesidad de crear una instancia de la clase):

- cantTotalPersonas (estático) en la clase Persona
- nro (estático) en la clase Domicilio

Variables de instancia (no estáticas, son variables de instancia que se declaran dentro de la clase y solo pueden ser accedidas a través de una instancia de la clase):

- id en las clases Persona y Domicilio
- nombreApellido en la clase Persona
- domicilio en la clase Persona
- dni en la clase Persona
- fechaNac en la clase Persona
- calle en la clase Domicilio
- loc en la clase Domicilio

Locales (son aquellos que se declaran dentro de un método y solo pueden ser accedidos dentro del mismo):

- edad en el método getEdad() de la clase Persona
- fN en el método getEdad() de la clase Persona

Ejercicio 8: Sea el siguiente ejercicio escrito en Pascal

```
1- Program Uno;
2- type tpuntero= ^integer;
3- var mipuntero: tpuntero;
4- var i:integer;
5- var h:integer;
6- Begin
7- i:=3;
8- mipuntero:=nil;
9- new(mipuntero);
10- mipunterno^:=i;
11- h:= mipuntero^+i;
12- dispose(mipuntero);
13- write(h);
14- i:= h- mipuntero;
15- End.
```

Variable	L-valor	Tiempo de vida	Alcance
i	automática	1-15	5-15
h	automática	1-15	6-15
Mipuntero	Automática?	9-15	4-15
^mipuntero	Dinámica?	9-12	10-12
Programa ppal	Automática?	1-15?	7-15

a) Indique el rango de instrucciones que representa el tiempo de vida de las variables i, h y mipuntero.

b) Indique el rango de instrucciones que representa el alcance de las variables i, h y mipuntero.

c) Indique si el programa anterior presenta un error al intentar escribir el valor de h. Justifique
d) Indique si el programa anterior presenta un error al intentar asignar a i la resta de h con mipuntero. Justifique
e) Determine si existe otra entidad que necesite ligar los atributos de alcance y tiempo de vida para justificar las respuestas anteriores. En ese caso indique cuál es la entidad y especifique su tiempo de vida y alcance.
f) Especifique el tipo de variable de acuerdo a la ligadura con el l-valor de las variables que encontró en el ejercicio.

c- no pq el dispose se hace después
d- sí porque se hace se hace antes
e- El programa ppal?

Ejercicio 9: Elija un lenguaje y escriba un ejemplo:

a. En el cual el tiempo de vida de un identificador sea mayor que su alcance
b. En el cual el tiempo de vida de un identificador sea menor que su alcance
c. En el cual el tiempo de vida de un identificador sea igual que su alcance

```c
#include <stdio.h>

void suma(){
    static int num = 0;
    num++;
    printf ("El valor de num es: %d\n",num);
}

int main(void) {
    int i;
    for (i = 1; i <= 3; i++) {
        suma();
    }
    return 0;
}
```

Se imprime en consola:

```
El valor de num es: 1
El valor de num es: 2
El valor de num es: 3
```

En el anterior programa, el alcance de num es solo dentro de la función suma, pero su tiempo de vida es desde la primera vez que se ejecuta la función suma hasta que finaliza el programa ya que se trata de una variable estática.

Ejemplo en Pascal:

```
1  program ejemploB;
2
3  type tpuntero = ^integer;
4
5  var
6  mipuntero: tpuntero;
7  begin
8     new(mipuntero);
9     mipuntero^ := 10;
10    dispose(mipuntero);
11    writeln(mipuntero^);
12 end.
```

En el anterior programa, el alcance de mipuntero desde la línea 6 (su declaración) hasta la 12, pero su tiempo de vida es desde la 8 hasta la 10 (su tiempo de vida termina por el dispose()). Después de liberar la memoria asignada a mipuntero, el puntero deja de existir y no se puede acceder al entero apuntado por él, como se intenta hacer en la línea 11.

Ejemplo en Python:

```
1  def ejercicioC():
2      x = 5
3      print(x)
4      y = 10
5      if x == 5:
6          z = 15
7          print(y, z)
8      print(y)
9
10 ejercicioC()
```

En el anterior programa, el alcance de x, y, y z está limitado a la función ejercicioC(). Además, su tiempo de vida coincide con el tiempo de ejecución de la función (desde que se ejecuta la función hasta que termina la función). Por lo tanto, el alcance y el tiempo de vida de x, y, y z son iguales.

Ejercicio 10: Si tengo la siguiente declaración al comienzo de un procedimiento:

int c; **en C**

var c:integer; **en Pascal**

c: integer; **en ADA**

Y ese procedimiento NO contiene definiciones de procedimientos internos. ¿Puedo asegurar que el alcance y el tiempo de vida de la variable "c" es siempre todo el procedimiento en donde se encuentra definida?. Analícelo y justifique la respuesta, para todos los casos.

En resumen, en los tres lenguajes mencionados, el alcance y el tiempo de vida de la variable c se limitan al procedimiento en el que se define, y no se extienden más allá de él. Esto es válido siempre y cuando no haya definiciones de procedimientos internos dentro del procedimiento principal, ya que en ese caso, la situación podría cambiar dependiendo de la definición y ubicación de las variables.

Ejercicio 11: a) Responda Verdadero o Falso para cada opción. El tipo de dato de una variable es?

I) Un string de caracteres que se usa para referenciar a la variable y operaciones que se pueden realizar sobre ella.

II) Conjunto de valores que puede tomar y un rango de instrucciones en el que se conoce el nombre.

III) Conjunto de valores que puede tomar y lugar de memoria asociado con la variable.

IV) Conjunto de valores que puede tomar y conjunto de operaciones que se pueden realizar sobre esos valores. verdadero

b) Escriba la definición correcta de tipo de dato de una variable.

- Definición:
  - Conjunto de valores
  - Conjunto de las operaciones
- Antes de que una variable pueda ser referenciada debe ligársele un tipo
  - Protege a las variables de operaciones no permitidas

Chequeo de tipos: verifica el uso correcto de las variables

Ejercicio 12: Sea el siguiente programa en ADA, completar el cuadro siguiente indicando para cada variable de que tipo es en cuanto al momento de ligadura de su l-valor, su r-valor al momento de alocación en memoria y para todos los identificadores cuál es su alcance y cual es su el tiempo de vida.
Indicar para cada variable su r-valor al momento de alocación en memoria

```
1.      with text_io; use text_io;
2.      Procedure Main is;
3.      type vector is array(integer range <>);
4.      a, n, p:integer;
5.      v1:vector(1..100);
6.      c1: constant integer:=10;
7.      Procedure Uno is;
1.  type puntero is access integer;
2.  v2:vector(0..n);
3.  c1, c2: character;
4.  p,q: puntero;
5.  begin
      7.5.1. n:=4;
      7.5.2. v2(n):= v2(1) + v1(5);
      7.5.3. p:= new puntero;
      7.5.4. q:= p;
      7.5.5. .....
      7.5.6. free p;
      7.5.7. .....
      7.5.8. free q;
      7.5.9. .....
7.6. end;
8. begin
9.  n:=5;
10. .....
11. Uno;
12. a:= n + 2;
13. .....
14. end
```

Ident.	Tipo	r-valor	Alcance	T.V.
a (linea 4)	automática	basura	4-14	1-14

→ ¿p, desde la siguiente no sería 5 ??

Identificador	Tipo	r-valor	Alcance	Tiempo de vida
a(línea 4)	automática	basura	5-14	2-14
n(línea 4)	automática	basura	5-14	2-14
p(línea 4)	automática	basura	5-14	2-14
V1(línea 5)	automática	basura	6-14	2-14
Main(2)	-	-	3-14	2-14
Uno(7)	-	-	1(8)-14	7-7.6
C1(3)	automática	basura	4-7.6	7-7.6
C2(3)	automática	basura	4-7.6	7-7.6
p(4)	automática	null	5-7.6	7-7.6
q(4)	automática	null	5-7.6	7-7.6
p^(7.5.3)	Dinámica	Basura	5-7.6	7.5.4- 7.5.6
q^(7.5.4)	dinámica	L-valor de p	5-7.6	7.5.4-7.6

Aclaración:
Ident.= Identificador / **Tipo** es el tipo de la variable respecto del l-value
T.V. = Tiempo de Vida / **r-valor** debe ser tomado al momento de la alocación en memoria.
El alcance de los identificadores debe indicarse desde la línea siguiente a su declaración.

```
ARCHIVO1.C
1.    int v1;
2.    int *a;
3.    Int fun2 ()
4.    { int v1, y;
5.      for(y=0; y<8; y++)
6.      { extern int v2;
7.        ...}
8.    }
9.    main()
10.   {static int var3;
11.   extern int v2;
12.   int v1, y;
13.   for(y=0; y<10; y++)
14.   { char var1='C';
15.    a=&v1;}
16.   }
ARCHIVO2.C
17.   static int aux;
18.   int v2;
19.   static int fun2( )
20.   { extern int v1;
21.     aux=aux+1;
22.     ...
23.   }
24.   int fun3( )
25.   { int aux;
26.     aux=aux+1;
27.     ...
28.   }
```

Identificador	Tipo	r-valor	Alcance	Tiempo de vida
V1(línea 1)	automática	0	2-4, 9-12, 17-20, 24-28	1-28
a(línea 2)	automática	Null?	3-16	1-28
*a(2)	dinámica	basura	3-16	15-16
Fun2(línea 3)	-	-	4-16	1-28
V1(4)	automática	basura	5-8	3-8
y(4)	Automática	basura	5-8	3-8
V2(6)	?	?	7	5-7
main(9)	-	-	10-16	1-28
Var3(10)	estática	0	11-16	<1-28>
V2(11)	automática	basura	12-16	?
V1(12)	automática	basura	13-16	9-16
y(12)	automática	basura	13-16	9-16
Var1(14)	Automática	'C'	15	13-16
Aux(17)	Estática	0	18-25	<1-28>
V2(18)	Automática	Basuta	19-28	1-28
Fun2(19)	-	-	20-28	<1-28>
V1(20)	?	?	21-23	19-23
Fun3(24)	-	-	25-28	1-28
aux	Estática	0	26-28	24-28