Práctico 4
Derivación de Programas Imperativos

Algoritmos y Estructuras de Datos I
2do cuatrimestre 2025

Laboratorio 1 (swap) Traducir e implementar en lenguaje C el siguiente programa que intercambia los valores de dos variables `x` e `y` de tipo `Int`, verificado en el práctico anterior.

```
z := x;
x := y;
y := z;
```

Laboratorio 2 (Asignaciones múltiples) Considerar las siguientes asignaciones múltiples

```
{Pre: x = X, y = Y}
x, y := x + 1, x + y
{Post: x = X + 1, y = X + Y}
```

```
{Pre: x = X, y = Y, z = Z}
x, y, z := y, y + x + z, y + x
{Post: x = Y, y = Y + X + Z, z = Y + X}
```

a) Para cada uno escribir un programa equivalente que sólo use secuencias de asignaciones simples.

b) Verificar que los programas propuestos en el punto anterior son correctos.

c) Traducir los programas resultantes a C, respectivamente.

Recordar: Como C no tiene asignaciones múltiples, siempre será necesario traducirlas primero a secuencias de asignaciones simples.

Laboratorio 3 (vocales) Crear un archivo `vocales.c` que contenga la función:

```
bool es_vocal(char letra)
```

que dado el caracter `letra` devuelve `true` si es una vocal y `false` en caso contrario. En la función `main` se le debe solicitar al usuario que ingrese un caracter y luego se debe mostrar un mensaje que indique si dicho caracter es una vocal o no según el resultado de la función `es_vocal()`. Tener en cuenta vocales mayúsculas y minúsculas.

NOTA: Definir una función que pida un caracter análoga a `pedir_entero()` pero para el tipo `char`.

NOTA: Recordar usar `%c` en vez de `%d` en el uso de `scanf()` y `printf()` para obtener / mostrar caracteres al usuario.

1. Para cada uno de los siguientes problemas, especificá utilizando pre y post condición y, derivá un programa que lo resuelva o definí un programa y demostrá que el programa es correcto.

   a) (Mínimo) Calcular el mínimo entre dos variables enteras `x` e `y`.

   b) (Valor Absoluto) Calcular el valor absoluto de un número entero.

Laboratorio 4 Traducir a Lenguaje C los programas definidos en el ejercicio 1. Incorporar al programa las pre y post condiciones utilizando el comando `assert`. En todos los casos el programa en C, debe solicitar los valores de las variables de entrada, e imprimir el resultado para que lo pueda ver el usuario.

2. Especificar y derivar un programa que calcule el factorial de un número.

3. (Función suma_hasta) Dado $N \geq 0$ queremos un programa que devuelva la suma de los primeros n naturales. Especificá el problema utilizando pre y post condición y, derivá un programa que lo resuelva o definí un programa y demostrá que el programa es correcto.

Laboratorio 5 Definir en C una programa que contenga la función

```
int suma_hasta(int n)
```

que toma un número entero `n` como argumento, y devuelve la suma de los primeros `n` naturales (ver ejercicio 3). En la función `main` pedir al usuario que ingrese el entero `n`, si es negativo imprimir un mensaje de error, y si es no negativo imprimir el resultado devuelto por `suma_hasta`.

Laboratorio 6 (Arreglos, entrada-salida) Escribir un programa que solicite el ingreso de un arreglo de enteros `int a[]` y luego imprime por pantalla. El programa debe utilizar dos nuevas funciones además de la función `main`:

- una que dado un tamaño máximo de arreglo y el arreglo, solicita los valores para el arreglo y los devuelve, guardándolos en el mismo arreglo `int a[]`; función con prototipo (también conocido como signatura o firma):

  ```
  void pedir_arreglo(int n_max, int a[])
  ```

- otra que imprime cada uno de los valores del arreglo `int a[]`, de prototipo:

  ```
  void imprimir_arreglo(int n_max, int a[])
  ```

4. (Suma de los elementos de un arreglo) Dado un arreglo de enteros, especificar y derivar un programa que calcule la suma de todos los elementos del arreglo.

Laboratorio 7 (Arreglos, Función sumatoria). Hacer un programa en un archivo con nombre `sumatoria.c` que contenga la función

```
int sumatoria(int tam, int a[])
```

que recibe un tamaño máximo de arreglo y un arreglo como argumento, y devuelve la suma de los elementos del arreglo. En la función `main` pedir los datos del arreglo al usuario asumiendo un tamaño constante previamente establecido (en tiempo de compilacilación).

5. Sea A un arreglo de enteros.

   a) Especificar y derivar un programa que determine si todos los elementos de A son mayores a 0.

   b) Especificar y derivar un programa que determine si algún elemento de A es mayor a 0.

Laboratorio 8 (Arreglos, todos positivos, algunos positivos). Hacer un programa en un archivo con nombre `positivos.c` que contenga las funciones

```
bool todos_pos(int tam, int a[])
bool algunos_pos(int tam, int a[])
```

que implementan los algoritmos derivados en los ejercicios 5a y 5b, respectivamente. En la función `main` pedir los datos del arreglo al usuario asumiendo un tamaño constante previamente establecido (en tiempo de compilación), e imprmir el mensaje "son todos positivos", en caso de que todos sean positivos, "alguno es positivo" en caso de que no todos, pero alguno sea positivo y "ninguno es positivo", en caso de que ninguno sea positivo.

6. Especificar y derivar un programa que calcule la suma de los elementos pares de un arreglo de enteros.

7. Dado un arreglo $A : array[0, N)\ of\ Num$ con $N \geq 0$, contar cuántas veces coinciden dos elementos:

   Const $N : Int, A : array\ [0, N)\ of\ Int$;
   Var $r : Int$;
   $\{P : N \geq 0\}$
   S
   $\{Q : \ r = \langle N\, i, j \ : 0 \leq i < j < N : \ A.i = A.j \,\rangle\}$

8. Dado un arreglo $A : array[0, N)\ of\ Num$ con $N \geq 0$, determinar si hay dos elementos que suman 8:

   Const $N : Int, A : array\ [0, N)\ of\ Int$;
   Var $r : Bool$;
   $\{P : N \geq 0\}$
   S
   $\{Q : \ r = \langle \exists\, i, j \ : 0 \leq i < j < N : \ A.i + A.j = 8 \,\rangle\}$

9. (Algoritmo de la división) Dados dos números, hay que encontrar el cociente y el resto de la división entera entre ellos.

   Ayuda: Para enteros $x \geq 0$ e $y > 0$, el cociente q y el resto r de la división entera de x por y están caracterizados por $x = q * y + r \wedge 0 \leq r \wedge r < y$. Por lo tanto, debemos derivar un programa S que satisfaga

   Const $x, y : Int$;
   Var $q, r : Int$;
   $\{P : \ \ x \geq 0 \wedge y > 0\}$ (precondición)
   S
   $\{Q : \ \ x = q * y + r \wedge 0 \leq r \wedge r < y\}$ (postcondición)

Laboratorio 9

(Algoritmo de la división) Crear un archivo llamado `division.c` que contenga la siguiente función:

```
struct div_t division(int x, int y){
...
}
```

donde la estructura `div_t` se define como

```
struct div_t {
  int cociente;
  int resto;
};
```

Esta función recibe dos enteros no negativos (divisor no nulo) y devuelve el cociente junto con el resto de la división entera. En la función `main` pedir al usuario los dos números enteros, imprimir un mensaje de error si el divisor es cero, o imprimir tanto el cociente como el resto en otro caso.

10. Especificar y derivar: Dado un arreglo $a : array[0, N)\ of\ Num$ con $N \geq 0$ determinar si alguno de sus elementos es igual a la suma de los anteriores. Usar fortalecimiento de invariante.

11. Especificar y derivar: Dado un arreglo $a : array[0, N)\ of\ Num$ con $N \geq 0$ determinar si sus elementos son iguales al factorial de la posición. Usar fortalecimiento de invariante.

12. Especificar y derivar un programa imperativo que calcule Fibonacci de un número dado. Usar fortalecimiento de invariante.

Laboratorio 10 (Fibonacci). Hacer un programa en un archivo con nombre `fibonacci.c` que contenga la función derivada en el punto 12.

```
int fibonnaci(int n)
```

En la función `main` pedir un entero y mostrar en pantalla el resultado de la función.

13. (Máxima diferencia) Dado un arreglo de enteros, calcular la máxima diferencia entre dos de sus elemento (en orden, el primero menos el segundo).

    La especificación del programa es:

    Const $N : Int$;
    Var $a : array[0, N)\ of\ Int; r : Int$;
    $\{P : N \geq 2\}$
    S
    $\{Q : r = \langle \text{Max}\ p, q\ : 0 \leq p < q < N :\ a.p - a.q \rangle\}$

14. (Segmento de suma máxima) Dado un arreglo de enteros, calcular la suma del segmento de suma máxima del arreglo.

    La especificación del programa es:

    Const $N : Int$;
    Var $a : array[0, N)\ of\ Int; r : Int$;
    $\{P : N \geq 0\}$
    S
    $\{Q : r = \langle \text{Max}\ p, q\ : 0 \leq p \leq q \leq N :\ sum.p.q \rangle$
    $[\![sum.p.q = \langle \sum i\ : p \leq i < q :\ a.i \rangle]\!]\}$

15. Dada la siguiente especificación:

    Const $M : Int, A : array[0, M)\ of\ Int$;
    Var $r : Int$;
    $\{P : M \geq 0\}$
    S
    $\{Q : r = \langle N\, p, q\ : 0 \leq p < q < M :\ A.p * A.q \geq 0 \rangle\}$

    Decir en palabras qué hace el programa y derivarlo.

16. Sea $N \geq 0$, especificar y derivar un programa que calcule el menor natural x que satisface $x^3 + x \geq N$.

    Ayuda: Especifique el problema (con pre y poscondición) de forma que en la poscondición queden conjunciones y asi poder utilizar la técnica "tomar término de la conjunción". Para ellos fijarse como se hace esto al especificar el Ejemplo 19.2 del libro.

17. Sea $N \geq 0$, especificar y derivar un programa que calcule el mayor natural x que satisface $x^3 + x \leq N$.

Ejercicios de laboratorio

Laboratorio 11 (Figuras). Se define el tipo `figura_t` como sigue:

```
typedef enum _forma {
    TRIANGULO,
    CUADRADO,
    CIRCULO
} forma_t;
typedef enum _color {
    ROJO,
    AZUL,
    VERDE,
    AMARILLO
} color_t;
typedef struct _figura {
    forma_t forma;
    color_t color;
    int tamano;
} figura_t;
```

a) Definir las funciones `es_roja`, `es_verde`, `es_azul`, `es_amarilla`, `es_triangulo`, `es_cuadrado`, `es_circulo`, que devuelva bool si la figura es de la característica correspondiente.

```
bool es_roja(figura_t f);
```

b) Definir la funcion `tam`, que devuelve el tamaño de una figura.

```
int tam(figura_t f);
```

c) Definir la función `todas_rojas`, que toma un entero `n` y un arreglo de figuras de tamaño `n` y devuelve verdadero si son todas rojas, y falso en caso contrario.

```
bool todas_rojas(unsigned int longitud, figura_t arr[]);
```

para ello, implementar el siguiente algoritmo.

Const $N : Int, A : array\ [0, N)\ of\ Figura$;
Var $r : Bool$;
$\{P : M \geq 0\}$
r, n := True, 0
do n $\neq$ N $\rightarrow$
  r, n := r $\wedge$ (es_roja(A[n]), n + 1
od
$\{Q : r = \langle \forall i\ : 0 \leq i < N :\ es\_roja(A.i) \rangle\}$

d) Hacer una función `main` inicializada de la siguiente manera

```
int main(void) {
    figura_t f1 = {.forma=Triangulo, .color=Rojo, .tamano=5};
    figura_t f2 = {.forma=Circulo, .color=Rojo, .tamano=1};
    figura_t f3 = {.forma=Triangulo, .color=Rojo, .tamano=4};
    unsigned int longitud = 3;
    figura_t arr[] = {f1, f2, f3};
    /* COMPLETAR */
    return 0;
}
```

que imprima en pantalla el mensaje "son todas rojas", o "no son todas rojas", según corresponda.

e) Derivar e implementar la función `triangulos_amarillos`, que toma un entero `n` y un arreglo de figuras de tamaño `n` y devuelve verdadero si son todos los triangulos son amarillos, y falso en caso contrario.

```
bool triangulos_amarillos(unsigned int longitud, figura_t arr[]);
```

f) Derivar e implementar la función `suma_tam`, que toma un entero `n` y un arreglo de figuras de tamaño `n` y devuelve la suma de los tamaños de cada figura.

```
int suma_tam(unsigned int longitud, figura_t arr[]);
```

g) Incorporar a la función `main` otro arreglo de figuras y mensajes mostrando los resultados de evaluar las funciones definidas para los diferentes arreglos.

Laboratorio 12 (Persona). Se define el tipo `persona_t` como sigue:

```
typedef struct_persona {
    char *nombre;
    int edad;
    float altura;
    float peso;
} persona_t;
```

Derivar e implementar las siguientes funciones:

```
float peso_promedio(unsigned int longitud, persona_t arr[]);
persona_t persona_de_mayor_edad(unsigned int longitud, persona_t arr[]);
persona_t persona_de_menor_altura(unsigned int longitud, persona_t arr[]);
```

Las tres funciones toman como argumento una longitud máxima de arreglo y un arreglo de personas. Devuelven respectivamente el promedio de peso, la persona de mayor edad y la persona de menor altura que se encuentra en el arreglo. Ayuda: para el promedio, es necesario derivar la suma primero.

Para probar las funciones, hacer una función `main` como la siguiente:

```
int main(void) {
    persona_t p1 = {.nombre="Paola", .edad=21, .altura=1.85, .peso=75};
    persona_t p2 = {.nombre="Luis", .edad=54, .altura=1.75, .peso=69};
    persona_t p3 = {.nombre="Julio", .edad=40, .altura=1.70, .peso=80};
    unsigned int longitud = 3;
    persona_t arr[] = {p1, p2, p3};
    printf("El peso promedio es %f\n", peso_promedio(longitud, arr));
    persona_t p = persona_de_mayor_edad(longitud, arr);
    printf("El nombre de la persona con mayor edad es %s\n", p.nombre);
    p = persona_de_menor_altura(longitud, arr);
    printf("El nombre de la persona con menor altura es %s\n", p.nombre);
    return 0;
}
```

Ejercicios extra

18. Derive un programa para calcular el máximo común divisor entre dos enteros positivos. Utilice la siguiente especificación:

    Const $X, Y : Int$;
    Var $x, y : Int$;
    $\{X > 0 \land Y > 0 \land x = X \land y = Y\}$
    S
    $\{x = mcd.X.Y\}$

    Utilice como invariante $\{I : x > 0 \land y > 0 \land mcd.x.y = mcd.X.Y\}$.

    Para la derivación serán de utilidad las siguientes propiedades del mcd:

    a) $mcd.x.x = x$
    b) $mcd.x.y = mcd.y.x$
    c) $x > y \;\Rightarrow\; mcd.x.y = mcd.(x - y).y$
    d) $y > x \;\Rightarrow\; mcd.x.y = mcd.x.(y - x)$

19. Considere las siguientes definiciones recursivas de la función de exponenciación $exp.x.y$, especificada como $exp.x.y = x^y$:

    a) Definición de complejidad lineal:

    $$\begin{array}{lll} exp.x.y & = & (\;\; y = 0 \to 1 \\ & & \Box\; y \neq 0 \to x * exp.x.(y-1) \\ & &) \end{array}$$

    b) Definición de complejidad logarítmica:

    $$\begin{array}{lll} exp.x.y & = & (\;\; y = 0 \to 1 \\ & & \Box\; y \neq 0 \to (\;\; y \bmod 2 = 0 \to exp.(x*x).(y \div 2) \\ & & \qquad\quad \Box\; y \bmod 2 = 1 \to x * exp.x.(y-1) \\ & & \qquad\quad) \\ & &) \end{array}$$

    Derive **dos** programas imperativos que calculen la exponenciación, cada uno utilizando una de las definiciones recursivas. Utilice la siguiente especificación:

Const $X, Y : Int$;
Var $x, y, r : Int$;
$\{x = X \land y = Y \land x \geq 0 \land y \geq 0\}$
S
$\{r = X^Y\}$

Utilice como invariante $\{I : y \geq 0 \land r * x^y = X^Y\}$.

20. Considere el ejercicio 5.

    a) ¿En el programa derivado en el ejercicio 5a se recorre todo el arreglo? Si la respuesta es afirmativa piense si esto es necesario. Si no lo es busque una manera de derivar un programa equivalente que no recorra innecesariamente todo el arreglo.

    b) Repita el mismo razonamiento sobre el ejercicio 5b y derive si es necesario el programa mejorado.

21. Derivar el siguiente programa

Const $M : Int, A : array\ [0, M)\ of\ Int$;
Var $r : Int$;
$\{P : M \geq 0\}$
S
$\{Q : r = \langle \mathrm{N}\, i\ : 0 \leq i < M :\ \langle \sum j\ : 0 \leq j < i :\ A.j \rangle \leq i * A.i \rangle\}$

22. Derivar el siguiente programa

Const $N : Int$;
Var $a : array\ [0, N)\ of\ Int$;
$r : Int$;
$\{P : N > 0\}$
S
$\{Q : r = \langle \mathrm{Max}\ i\ : 0 \leq i < N \land \langle \forall j\ : 0 < j < i :\ a.(j-1) < a.j \rangle :\ i \rangle\}$

Nota: Antes de derivar decir que debería hacer el programa.

Nota: No se puede usar ∞ en el programa.

23. (Máxima diferencia cuadrada) Derivar un programa para la siguiente especificación:

Const $N : Int$;
Var $a : array[0, N)\ of\ Int; r : Int$;
$\{P : N \geq 2\}$
S
$\{Q : r = \langle \mathrm{Max}\ p, q\ : 0 \leq p < q < N :\ (a.p - a.q)^2 \rangle\}$