Trabajo Práctico Nº 5
Funciones y Procedimientos

Objetivo: Utilizar en los pseudocódigos las herramientas de modularización de programas y comunicación entre módulos

Hemos visto que al construir programas resulta muy útil pensar que el problema puede descomponerse en subproblemas a partir de cuyas soluciones obtener una solución al problema original. Hasta ahora, eso nos sirvió para darle forma al código de nuestros programas en segmentos que podían ser bien diferenciados (usando comentarios). Sin embargo, en ocasiones esos segmentos pueden ser un poco largos y llevar a que al ver el programa en su conjunto, se pierda de vista la distinción y el código se vuelve confuso. Esto se nota particularmente bien en programas de mediana o gran envergadura, aunque puede ocurrir también en algunos más pequeños. Así, los *procedimientos* y *funciones* (en general *subprogramas*) vienen a ayudarnos a organizar la programación, pues nos permiten que esos segmentos de código que podían resultar confusos se codifiquen por separado con algunas ventajas adicionales:

- se les asocia un nombre, por lo que si el segmento de código debía repetirse en diferentes partes del programa, ahora puede reemplazarse todo el segmento por su nombre en cada uno de los lugares donde antes se repetía el segmento completo, permitiendo además que el código termine siendo más breve y por tanto más ágil para la lectura; en el lugar donde está el nombre del subprograma se dice que hay una *invocación* al mismo;

- en ocasiones, las repeticiones no eran totalmente idénticas, sino que había ciertos datos que cambiaban de una a otra: a los subprogramas se les pueden asociar *parámetros*, es decir datos que pueden cambiar de valor de una invocación a otra; los valores que se asocian con los parámetros en cada invocación del subprograma suelen conocerse como *argumentos*, y la correspondencia se da por cantidad y posición: el primer argumento que se indica en la invocación corresponde al valor que se asigna al primer parámetro del procedimiento, el segundo argumento corresponde al valor asignado al segundo parámetros, y así hasta el último, por lo que debe haber tantos argumentos en la invocación como parámetros se definieron en la declaración del subprograma.

La introducción de subprogramas en la codificación de los programas lleva a introducir algunos conceptos relacionados con el uso de variables: los subprogramas tienen la misma estructura interna que un programa completo, de modo que pueden declararse variables que ayuden a lograr lo que se pretende pero que solo sirven para ese subprograma: son variables *locales* al subprograma, y tienen la característica fundamental de que los valores que tomen sólo se conocen dentro del subprograma. Por su parte, las que se declaran a nivel del programa principal (el único que codificábamos hasta ahora, son *globales* para los subprogramas: cualquier subprograma puede hacer uso de ellas y cualquier cambio que un subprograma (o el principal) haga a una de ellas, afectará al resto de los subprogramas que la utilicen o al mismo programa principal en cualquier ejecución posterior al cambio.

El subprograma (o el programa principal) que hace una invocación a otro subprograma se conoce como el *llamador*.

Los parámetros de un subprograma pueden ser de dos clases, por ahora: *por valor* o *por referencia*. Los parámetros por valor son variables locales al procedimiento cuyo valor se asigna en la invocación, y por ser variables locales su cambio no se nota fuera del subprograma. Los parámetros por valor constituyen datos de entrada para el subprograma y los argumentos correspondientes pueden estar dados como constantes o variables. En cambio, los parámetros por referencia sólo pueden estar asociados con argumentos variables, pues cualquier alteración de los mismos en el subprograma afectará el valor asociado a la variable argumento del llamador; la principal utilidad de estos parámetros es la de poder hacer referencia a dichos valores con un nombre que resulta significativo dentro del subprograma, independientemente de como se lo conozca en el resto. Los parámetros por referencia, pueden ser de entrada o de salida para el subprograma.

Como se dijo, hay dos clases de subprogramas: los procedimientos y las funciones:

- Las funciones sólo admiten parámetros por valor y siempre tienen un valor asociado (el *resultado*), pues están pensadas para ser utilizadas como parte de una expresión (igual que las funciones primitivas). Así, solo pueden devolver valores de tipos simples, como los que hemos utilizado hasta ahora.

- Los procedimientos admiten parámetros de ambas clases, por lo que podrían tener varias salidas. No pueden formar parte de expresiones, su invocación se realiza como si fuesen una instrucción más en el lenguaje.

En ocasiones, hay un grupo de subprogramas que pueden considerarse relacionados entre sí por algún criterio, pudiendo eventualmente compartir algunos datos (o no). Dichos subprogramas forman un *módulo* y aparecen con frecuencia al resolver problemas de mediana o gran envergadura.

El ejemplo que se presenta a continuación fue elaborado por el Profesor Marcos Santa Cruz:

Ejemplo:

En una competencia de autos, en la que participan 20 corredores como máximo (cada uno identificado por un número de 2 dígitos), se reciben los resultados de 3 pruebas clasificatorias.

Los datos recibidos de cada prueba, para cada corredor son: tiempo utilizado para terminar la prueba (expresado en horas, minutos y segundos), el número de identificación y la cantidad de giros completos realizados. Si un corredor que inició la prueba no la finaliza por algún problema, los valores del tiempo están en cero. Se informa antes de iniciar cada prueba, la cantidad total de corredores que interviene en la misma y los datos se reciben en el orden en que han clasificado.
Se debe informar al finalizar cada prueba: el mejor tiempo expresado en segundos, identificación del corredor, la diferencia de tiempo en segundos sobre el segundo clasificado
(si no llegó a terminar la prueba un segundo competidor es nula) y el porcentaje de corredores (con respecto a los que la iniciaron) que finalizaron la prueba a una distancia no mayor a dos giros de los recorridos por el ganador.
Luego de las pruebas clasificatorias se lleva a cabo la etapa final, de características similares a un prueba clasificatoria, recibiendo los mismos datos de entrada e igual orden que en una prueba clasificatoria. Se debe realizar un ajuste de tiempo utilizado por los corredores que accedieron al podio, restándole a cada uno, las diferencias obtenidas sobre los segundos clasificados en cada prueba en la que terminaron primero. De esa manera la información requerida luego del ajuste es cantidad de corredores que terminaron la etapa y la identificación, cantidad de giros realizados y el tiempo utilizado (expresado en horas, minutos y segundos) del ganador final de la carrera. Se descarta toda igualdad de tiempos registrado en los primeros tres lugares.

Análisis del Problema:

¿Qué Entradas se requieren (Tipo y Cantidad)?

a) Por cada una de las 3 pruebas:
a.1) Cantidad de corredores que inician la prueba
a.2) Datos obtenidos por cada uno de los corredores, informados en orden de llegada:
a.2.1) Número de corredor (valor entero de 2 dígitos)
a.2.2) Tiempo utilizado para terminar la prueba:
a.2.2.1) Cantidad de horas (valor entero de 2 dígitos)
a.2.2.2) Cantidad de minutos (valor entero de 2 dígitos)
a.2.2.3) Cantidad de segundos (valor entero de 2 dígitos)
a.2.3) Cantidad de giros completos realizados (valor entero de 2 dígitos)
b) De la etapa final:
b.1) Cantidad de corredores que inician la etapa final
b.2) Datos obtenidos por cada uno de los corredores, informados en orden de llegada:
b.2.1) Número de corredor (valor entero de 2 dígitos)

b.2.2) Tiempo utilizado para terminar la prueba:
b.2.2.1) Cantidad de horas (valor entero de 2 dígitos)
b.2.2.2) Cantidad de minutos (valor entero de 2 dígitos)
b.2.2.3) Cantidad de segundos (valor entero de 2 dígitos)
b.2.3) Cantidad de giros completos realizados (valor entero de 2 dígitos)
b.3) Número de identificación de cada corredor ganador en las pruebas y los segundos de diferencia sobre el segundo clasificado.

¿Cuál es la Salida deseada? (Tipo y Cantidad)

a) En cada una de las 3 pruebas:
a.1) Número de identificación del corredor ganador.
a.2) Tiempo utilizado por el ganador expresado segundos.
a.3) Diferencia de segundos sobre el segundo clasificado.
a.4) Porcentaje de corredores que finalizaron la prueba con un margen máximo de dos giros.
b) En la prueba final:
b.1) Número de identificación del corredor ganador final.
b.2) Tiempo utilizado por el ganador con el ajuste final, expresado en horas, minutos y segundos.
b.3) Cantidad de giros realizados por el ganador final.
b.4) Cantidad de corredores que terminaron la etapa final

¿Qué método produce la Salida deseada?
Se deben procesar dos tareas realizadas en diferentes tiempos con características y requerimientos particulares: las pruebas clasificatorias y la etapa final. La etapa final debe manejar además sus propios datos de entrada, resultados obtenidos durante las pruebas clasificatorias.

Programa En Pseudocódigo:

El pseudocódigo a continuación contiene algunos cambios respecto del aportado por el Prof. Santa Cruz. Principalmente, se le han agregado comentarios en las declaraciones de variables, se modificaron otros comentarios y se convirtió un procedimiento en una función.

```
PROGRAMA: CARRERA DE AUTOS
VARIABLES:
NG1, NG2, NG3 : ENTERO 2
DIFSEG1, DIFSEG2, DIFSEG3: ENTERO 5

PROCEDIMIENTO: PruebaCla(REF: NG : ENTERO 2; REF: Tseg : ENTERO 5)
VARIABLES: /* controla lo ocurrido en cada prueba clasificatoria */
THcor, TMcor, TScor // tiempo utilizado por el corredor
GIROS, // cantidad de giros
CANTcor, // cantidad de corredores
NROcor, // nro. De corredor
INDcor, // índice del Rep. PARA por corredor
MINGIROS, // número mínimo de giros para contar
SELEC // contador para el porcentaje
: ENTERO 2
TPRIMERO, // Tiempo en segundos del ganador
TSEGUNDO // Tiempo en segundos del segundo en llegar
: ENTERO 5
PORCGIROS: REAL 5,2
HACER
SELEC := 0
LEER: CANTcor
REPETIRPARA: INDcor := 1, CANTcor
LEER: NROcor, GIROS, THcor, TMcor, TScor
SI: (INDcor = 1) ENTONCES // es el primero
TPRIMERO := TiempoEnSeg(THcor, TMcor, TScor)
NG := NROcor
MINGIROS := GIROS -2
```

```
SINO
SI: (INDcor = 2) ENTONCES // es el segundo
SI: (THcor+TMcor+TScor>0 ) ENTONCES
TSEGUNDO := TiempoEnSeg(THcor, TMcor, TScor)
Tseg := TSEGUNDO – TPRIMERO // diferencia entre primero y segundo
SINO
Tseg := 0 // si no termina el segundo la diferencia es nula
FINSI
FINSI
FINSI
SI: (THcor+TMcor+TScor> 0) Y (GIROS >= MINGIROS) ENTONCES
SELEC := SELEC + 1 // contador para el porcentaje
FINSI
FIN REPETIR PARA
PORCGIROS:= SELEC * 100 / CANTcor
IMPRIMIR: “GANO EL CORREDOR “, NG,
“UTILIZANDO UN TIEMPO DE “, TPRIMERO,
“SEGUNDOS Y CON UNA DIFERENCIA SOBRE EL SEGUNDO”,
“CLASIFICADO DE “, TSEG, “ SEGUNDOS”
IMPRIMIR: “EL PORCENTAJE DE CORREDORES QUE FINALIZARON LA PRUEBA
DENTRO DE LOS DOS GIROS : “, PORCGIROS
FIN HACER
FIN PROCEDIMIENTO

PROCEDIMIENTO: EtapaFinal(VG1 : ENTERO 2; DIF1 : ENTERO 5; VG2 : ENTERO2;
DIF2 : ENTERO 5; VG3 : ENTERO 2; DIF3 : ENTERO 5)
// Este procedimiento controla lo que ocurre en la etapa final
// Los parámetros representan la información de entrada producida por las pruebas clasificatorias:
// VGn identifica al ganador y DIFn la diferencia con el segundo en la prueba n
VARIABLES:
THcor, TMcor, TScor, GIROS, CANTcor, NROcor, INDcor, // similares al proc. PruebaCla
SELEC // contador de corredores que finalizan la etapa final
// las restantes variables describen información de los 3 primeros em la etapa final
// y del que resulta ganador después de los ajustes
COR1, COR2, COR3, GANA, // identificación del corredor
GIROSGANA, GIROS1, GIROS2, GIROS3 // cantidad de giros
: ENTERO 2
SEG1, SEG2, SEG3, SEGGANA: ENTERO 5 // tiempo en segundos
HACER
LEER: CANTcor
SELEC := 0
REPETIR PARA: INDcor = 1, CANTcor
LEER: NROcor, GIROS, THcor, TMcor, Tscor
SI: (INDcor<=3) ENTONCES // se guardan los datos de los 3 primeros
CASO // ajustando los tiempos según las pruebas clasificatorias
(INDcor = 1) : // primer lugar en la etapa final
COR1:= NROcor
GIROS1:= GIROS
SEG1:=TiempoEnSeg(THcor, TMcor, TScor)
SEG1:= AJUSTE (VG1, DIF1, VG2, DIF2, VG3, DIF3, COR1, SEG1)
(INDcor = 2 ) : // segundo lugar en la etapa final
COR2 := NROcor
GIROS2:= GIROS
SEG2:=TiempoEnSeg(THcor, TMcor, TScor)
SEG2 := AJUSTE(VG1, DIF1, VG2, DIF2, VG3, DIF3, COR2, SEG2)
(INDcor = 3 ) : // tercer lugar en la etapa final
COR3 := NROcor
GIROS3 := GIROS
SEG3 := TiempoEnSeg(THcor, TMcor, TScor)
SEG3 := AJUSTE(VG1, DIF1, VG2, DIF2, VG3, DIF3, COR3, SEG3)
FINCASO
FIN SI
SI: ( THcor + TMcor + TScor> 0 ) ENTONCES
SELEC : = SELEC + 1 // se cuentan cuántos corredores terminaron
```

```
FIN SI
FIN REPETIR PARA
GANA := COR1
SEGGANA := SEG1
GIROSGANA := GIROS1
SI: (SEG2 <SEGGANA )ENTONCES /* reordena luego del ajuste final */
SI: (SEG2 < SEG3) ENTONCES
GANA := COR2
GIROSGANA := GIROS2
SEGGANA := SEG2
SINO
GANA := COR3
GIROSGANA := GIROS3
SEGGANA := SEG3
FINSI
SINO
SI: (SEG3 < SEGGANA) ENTONCES
GANA := COR3
SEGGANA := SEG3
GIROSGANA := GIROS3
FINSI
FINSI
TiempoEnHMS(SEGGANA, THcor, TMcor, TScor) // tiempo ganador en horas, minutos y segundos
IMPRIMIR: “EL GANADOR FINAL FUE EL CORREDOR “, GANA, “ EN “, THcor, “HORAS”,
TMcor, “MINUTOS”, TScor , “SEGUNDOS”, “ Y “, GIROSGANA, “GIROS”
IMPRIMIR: “TERMINARON LA ETAPA FINAL “, SELEC, “ CORREDORES.”
FIN HACER
FIN PROCEDIMIENTO

FUNCION: TiempoEnSeg (VH : ENTERO 2; VM : ENTERO 2; VS : ENTERO 2) : ENTERO 5
HACER /* convierte el tiempo de horas, minutos y segundos en segundos */
TiempoEnSeg := VH * 3600 + VM * 60 + VS
FIN HACER
FIN FUNCION

PROCEDIMIENTO: TiempoEnHMS(DATO : ENTERO 5; REF: H : ENTERO 2;
REF: M : ENTERO 2; REF: S : ENTERO 2)
HACER /* convierte el tiempo en segundos en cantidad de horas, minutos y segundos */
H := ENT(DATO / 3600) /* función primitiva parte entera */
DATO := DATO – H * 3600
M := ENT(DATO / 60)
S := DATO – M * 60
FIN HACER
FIN PROCEDIMIENTO

FUNCION: AJUSTE(DVG1 : ENTERO 2; DDIF1 : ENTERO 5; DVG2 : ENTERO2;
DDIF2 : ENTERO5; DVG3 : ENTERO 2; DDIF3 : ENTERO 5;
CORR : ENTERO 2; TsegCorr : ENTERO 5)
// Compara los datos de las pruebas clasificatorias con los del corredor en la etapa final
VARIABLES:
diferencia : ENTERO 5
HACER
CASO: (CORR)
DVG1 : diferencia := DDIF1
DVG2 : diferencia := DDIF2
DVG3 : diferencia := DDIF3
FIN CASO
Ajuste := TsegCorr – diferencia
FIN HACER
FIN PROCEDIMIENTO

HACER // PROGRAMA PRINCIPAL
PruebaCla(NG1, DIFSEG1)
```

PruebaCla(NG2, DIFSEG2)
PruebaCla(NG3, DIFSEG3)
EtapaFinal(NG1, DIFSEG1, NG2, DIFSEG2, NG3, DIFSEG3)
FIN HACER
FIN PROGRAMA

En los ejercicios que se plantean a continuación, tener en cuenta que las funciones deberían depender solo de los datos que reciben como parámetro, por lo que las lecturas necesarias deberían estar en el llamador. Esto suele ser deseable y esperable también para los procedimientos, aunque no en todos los casos. Analice cada situación cuidadosamente a la hora de escribir el código. En la medida de lo posible, discuta con sus compañeros.

Ejercicios sugeridos para trabajar en clase:

1. Codificar funciones para calcular:
   a. el máximo entre 2 números enteros
   b. el máximo entre 3 números enteros
   c. el máximo entre 5 números enteros
   d. el máximo entre 10 números enteros

En cada inciso donde sea posible, hacer dos versiones: una que calcule el máximo usando las funciones definidas en los incisos anteriores, y otra que no las use.
¿En qué cambiaría el código si se pidiera calcular el máximo entre números reales?

2. Diseñar una aplicación que ofrezca, a través de un menú, la posibilidad de calcular distancias entre puntos del plano o del espacio. Se deben codificar funciones que calculen las distancias a partir de las coordenadas de los puntos, que llegan como parámetros.

3. Diseñar una aplicación que ofrezca, a través de un menú, la posibilidad de calcular áreas de figuras geométricas específicas. Se deben codificar funciones que calculen las áreas de: círculo, cuadrado, rectángulo, triángulo y trapecio. *Las fórmulas pueden buscarse en libros de geometría de nivel secundario (polimodal)*.

4. Definir funciones o procedimientos, según convenga, para calcular:
   a. las raíces (reales o complejas) de una ecuación de 2do grado
   b. el promedio de 6 números enteros
   c. la conversión de temperaturas Celsius a Fahrenheit y viceversa
   d. el módulo de un vector del plano dadas sus coordenadas
   e. la suma de dos vectores dados del plano
   f. la conversión de letras mayúsculas a minúsculas y viceversa
   g. el valor del número PI usando la función primitiva ARCTAN

Ejercicios sugeridos para trabajar fuera del horario de clase:

5. Definir funciones que permitan convertir:
   a. un número entero binario a base decimal
   b. un número entero decimal a base 2
   c. un número entero en base B a decimal
   d. un número entero decimal a base B
   e. un número entero en base B a base C

6. Definir las siguientes funciones:
   a. **signo**, que decide si un número es positivo, negativo o cero (devuelve 1, -1, 0 respectivamente)
   b. **esMayus**, que decide si una letra dada es mayúscula o no lo es
   c. **esVocal**, si una letra dada es vocal o no lo es
   d. **esMultiplo**, que decide si el primer parámetro es múltiplo del segundo; ambos parámetros son números enteros

7. Simular 3 ejecuciones distintas del siguiente código, con valores diferentes para las variables G1, G2 y G3. Examinar el código y determinar su salida.

```
Programa Prueba
Variables
G1, G2, G3: Entero 3

ProcedimientoPP (pValor : Entero 3; REF: pRef : Entero 3)
Variables
L1, L2: Entero 3
Hacer
L1:= G2
L2:= pRef
pValor := L1 + L2
pRef := L1 – L2
G3 := 10 * L1
Fin Hacer
Fin Procedimiento
Hacer
Leer: G1, G2, G3
Imprimir: ' antes de PP, G1=' , G1
Imprimir: ' antes de PP, G2=' , G2
Imprimir: ' antes de PP, G3=' , G3
PP (G2, G1)
Imprimir: ' despues de PP, G1=' , G1
Imprimir: ' despues de PP, G2=' , G2
Imprimir: ' despues de PP, G3=' , G3
Fin Hacer
Fin Programa
```

8. Hacer un programa que imprima todos los números naturales primos menores que un número dado. *Recordar que los números naturales primos son los que sólo admiten 2 divisores: 1 y sí mismos, y que los divisores de un número distintos de él mismo son menores que la mitad.*