Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología
Sede: Oro Verde

FUNDAMENTOS DE PROGRAMACIÓN

RESUMEN DE CONTENIDOS N°7

Estructuras de Control: Secuencia y Selección

RESUMEN DE CONTENIDOS N° 7

Estructuras de Control: Secuencia y Selección

Introducción

Hasta aquí, se han desarrollado programas empleando tres acciones primitivas fundamentales: asignación, lectura y escritura. Las mismas han permitido resolver problemas sencillos, con la característica además, de que las acciones que lo formaban se ejecutaban secuencialmente, en el orden en que aparecían, es decir: una a continuación de otra.

Pero en el diseño de algoritmos, para la resolución de problemas más complejos, es necesario contar con herramientas que nos permitan modificar el orden lineal de ejecución de las acciones, donde el ejecutante pueda tomar decisiones y determinar qué acción realizar en el momento de ejecución del algoritmo.

Para ello, la diagramación estructurada nos brinda recursos propios. En rigor, esta metodología establece que todo algoritmo –por más complejo que sea – puede elaborarse mediante el uso de sólo tres estructuras lógicas de control:

- **Secuencia**
- **Selección**
- **Iteración**

Estas **estructuras** presentan la característica de tener un único punto de entrada y un único punto de salida. Un programa definido en base a estas estructuras, es más fácil de entender, y permite, por lo tanto, detectar los errores de lógica más rápidamente.

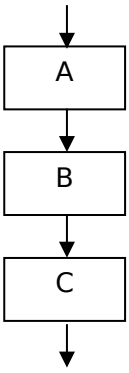
A continuación, se explicará cada una de las estructuras de control citadas, en algorítmica básica, y luego su equivalente en C++.

ESTRUCTURAS LÓGICAS DE CONTROL

SECUENCIA

Es un conjunto de acciones que se ejecutan en el mismo orden en que aparecen en el algoritmo, una a continuación de la otra.

La representación gráfica en un diagrama de flujo es:



Los bloques A, B, C representan una acción elemental o un módulo; cada uno de ellos, con un único punto de entrada y un único punto de salida.

En pseudocódigo o lenguaje de programación, la secuencia se especifica indicando las acciones en el orden en que deben ser ejecutadas, separadas entre sí por el signo de puntuación "**;**", (escritas en el mismo renglón o en diferente):

```
A; B;
C;
```

SELECCIÓN

La necesidad de contar con una estructura de **selección** que nos permita tomar decisiones en un algoritmo, se planteó en los primeros problemas, en los algoritmos no computacionales, donde se usaban condiciones que permitían al ejecutante decidir qué grupo de acciones ejecutar y cuáles no.

Ejemplo:

SI se tiene soda
ENTONCES agregar en la taza media cucharadita de soda
SINO agregar en la taza media cucharadita de agua fría
FINSI

Las estructuras de **selección** otorgan al ejecutante, la posibilidad de alterar el flujo lineal de control de las acciones de un algoritmo. Permiten tomar decisiones y elegir un camino a seguir en base a ciertos valores específicos del algoritmo, ya sean datos que se ingresan o resultados de cálculos que se realizan.

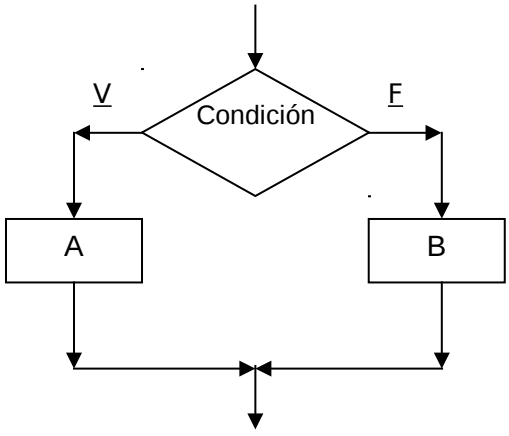
Observamos 2 tipos de estructuras de selección:

- condicional o de decisión
- selección múltiple

a. Estructura condicional o de decisión

Esta estructura permite elegir el camino a seguir según el valor de verdad de una condición.
La estructura **condicional** permite al ejecutante o procesador, elegir las acciones a seguir entre dos alternativas específicas, de acuerdo al valor de una condición en el momento de ejecución del algoritmo o programa.

La representación de esta estructura en un diagrama es:



donde **condición**, es cualquier elemento del ambiente del algoritmo que arroje un resultado lógico (**Verdadero** o **Falso**; **True** o **False**), es decir que el elemento **condición** puede ser una **variable o expresión lógica.**

Tanto **A** como **B** representan una acción primitiva o un conjunto de acciones.

La presencia de esta estructura en un programa, le indica al procesador que debe:

1) Evaluar la condición planteada
   a) si la condición toma un valor **Verdadero** (**True**), ejecutar el bloque A; y finaliza la estructura condicional indicada con el punto.
   b) si la condición toma un valor **Falso** (**False**), ejecutar el bloque B; y finaliza la estructura condicional indicada con el punto.
2) En ambos casos, luego continúa la ejecución del algoritmo con la acción posterior al punto.

La sintaxis en pseudocódigo sería:

```
SI condición
   ENTONCES    A
   SINO        B
FINSI
```

donde las palabras claves **SI** y **FINSI** indican, respectivamente, el comienzo y fin de la estructura de selección.

De acuerdo al valor de la condición, el procesador ejecuta A (alternativa verdadera) o B (alternativa falsa) y luego continúa con la acción que sigue al **FINSI**.

A esta estructura se la considera como una unidad, con un **único punto de entrada y un único punto de salida**. Su punto de entrada es la evaluación de la condición y su punto de salida, luego de haberse ejecutado el camino correspondiente, es la acción que sigue al punto (en el diagrama) o al **FINSI** (en pseudocódigo).

En C++, esta estructura es ***If-else***: Se evalúa la expresión lógica planteada a continuación del **if** y si es distinta de cero (verdadero) se realizan las acciones indicadas a continuación; si la expresión lógica es cero (falso), se realizan las acciones a continuación del **else**.

Sintaxis	Ejemplo

`if (expresión lógica)`
`  acción1;`
`else`
`  acción 2;`
	`int c=0;`
	`if ( c==200 )`
	`  c = c/2;`
	`else  c = 2*c;`

Ejemplo Práctico 1:
Una compañía de turismo ha definido una política de promoción de sus empresas para lo cual ha fijado descuentos para sus clientes en función de los viajes anteriores realizados.
El valor del descuento es del 20% para aquellos clientes que, considerando los viajes realizados en el último año y el que desean realizar, han recorrido más de 3000 Kms. y del 5% para aquellos que no han alcanzado dicha cifra.
Se desea realizar un algoritmo que calcule el importe a pagar por un cliente, si se ingresa su nombre y apellido, el total de kilómetros recorridos hasta el momento, y los datos del viaje a contratar: precio, destino, kilómetros a recorrer. Informar los datos ingresados, el monto del descuento y el monto a pagar por el cliente.

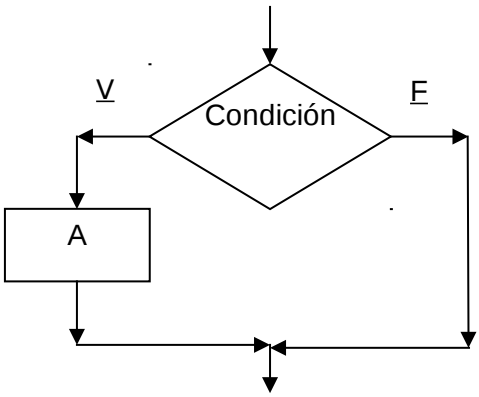
```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    string nya, destino;
    int kms, kmrec, total;
    float precio, desc, imp;

    cout<<"Ingrese los siguientes datos:"<<endl;
    cout<<"Nombre: ";
    getline(cin,nya);
    cout<<"Total Kms recorridos: ";
    cin>>kmrec;
    cout<<"Precio: $ ";
    cin>>precio;
    cin.get();
    cout<<"Destino: ";
    getline(cin,destino);
    cout<<"Kms a recorrer: ";
    cin>>kms;
    total=kmrec+kms;
    if(total>3000)
        desc=precio*0.20;
    else desc=precio*0.05;
    imp=precio-desc;

    cout<<endl; //linea en blanco
    cout<<"Cliente: "<<nya<<" Acumulados:  "<<kmrec<<endl;
    cout<<"Datos del viaje: $"<<precio<<" "<<kms<<" Kms."<<" "<<destino<<endl;
    cout<<"Descuento: $"<<desc<<" Importe: $"<<imp;
    return 0;
}
```

Seguimiento:
Datos: ‘Juan Pérez’, 3100, 350, ‘Misiones’, 800

NYA	KMREC	PRECIO	KMS	DESTINO	TOTAL	DESC	IMP
‘Juan Pérez’	3100	1000	800	‘Misiones’	3900	200	800

Salida por pantalla
Cliente: Juan Pérez Acumulados: 3100 Datos del Viaje: $1000 800 Kms. Misiones Descuento: $200 Importe: $800

Además, la estructura condicional brinda la posibilidad de plantear que por la alternativa falsa no se especifiquen acciones a ejecutar. De esta manera, en el condicional, la salida por **VERDADERO** deberá presentar **al menos una acción** a ejecutar, mientras que por el camino de la condición **FALSA** no se plantea ninguna acción. Esta variante de la estructura condicional se representa en diagrama de la siguiente manera:



En pseudocódigo, la sintaxis equivalente a este caso es:

SI condición
ENTONCES A
FINSI

En C++ la salida por **cero** (falso) puede obviarse; en tal caso, si la expresión arroja **cero** (falso) no se ejecutará acción alguna.

Sintaxis

```
if (expresión lógica)
      acción1;
```

Ejemplo Práctico 2:

Una compañía de turismo ha definido una política de promoción de sus empresas para lo cual ha fijado un descuento del 20% para aquellos clientes que, considerando los viajes realizados en el último año y el que desean realizar, superen los 3000 Kms. de recorrido.
Se desea realizar un algoritmo que calcule el importe a pagar por un cliente, si se ingresa su nombre y apellido, el total de Kms. recorridos hasta el momento, y los datos del viaje a contratar: precio, destino, Kms. a recorrer. Informar los datos ingresados, el monto del descuento y el monto a pagar por el cliente.

```
1   #include <iostream>
2   #include <string>
3   using namespace std;
4
5   int main() {
6       string nya, destino;
7       int kms, kmrec, total;
8       float precio, desc, imp;
9
10      ......
11      //Ídem a ejercicio práctico 1
12      if(total>3000)
13          desc=precio*0.20;
14      imp=precio-desc;
15      ......
16      //Ídem a ejercicio práctico 1
17
18      return 0;
19  }
```

Estructuras condicionales anidadas

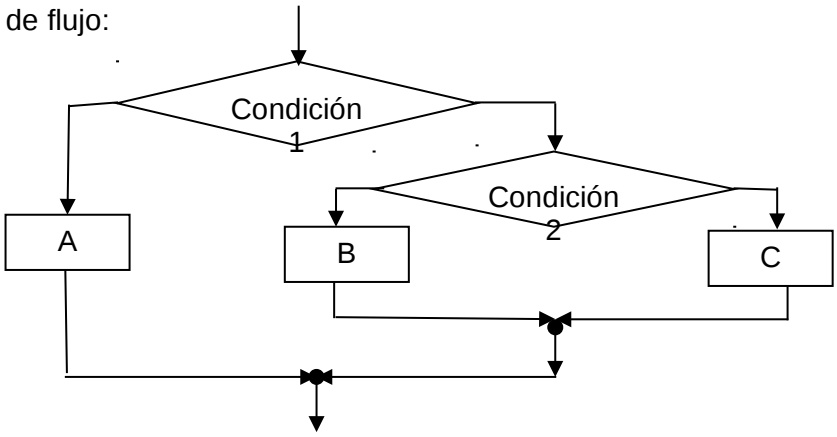
En una estructura condicional, tanto la alternativa verdadera como la falsa pueden presentar a su vez, otras estructuras condicionales, dando lugar a esquemas como el siguiente:
En pseudocódigo:

```
SI condición1
   ENTONCES   A
   SINO      SI condición2
                ENTONCES   B
                SINO   C
             FINSI
FINSI
```

En este esquema hablamos de condicionales anidados, con un condicional externo - el primero que se plantea (evalúa la condición1) - y condicionales internos (en este caso, aparece uno solo que evalúa la condición2).

Este esquema puede complicarse, si en A, B, y/o C planteamos otros condicionales internos, pero debe tenerse en cuenta que cada condicional debe corresponder al esquema **SI-FINSI**.

En diagrama de flujo:



En C++:

```
if (expresión_lógica 1)
{
  Acciones;
}
else if(expresión_lógica 2)
{
  Acciones;
} else if
...
...
} else
{
  Acciones;
}
```

Ejemplo Práctico 3:

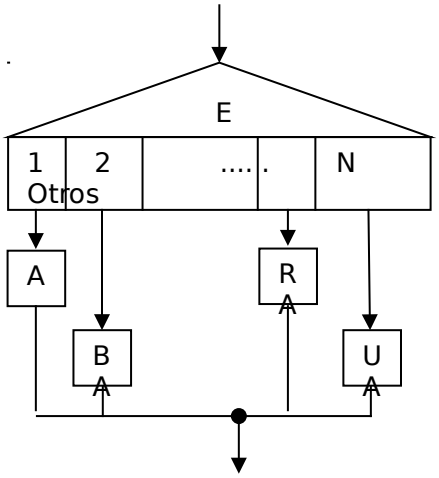
Una compañía de turismo ha definido una política de promoción de sus empresas para lo cual ha fijado descuentos para sus clientes en función de los viajes anteriores realizados. El valor del descuento es del 30% para aquellos clientes que, considerando los viajes realizados en el último año y el que desean realizar, han recorrido más de 3500 Kms., del 15 % si han sumado más de 2000 Kms., del 5% para aquellos que han superado los 500 kms. Se desea realizar un algoritmo que calcule el importe a pagar por un cliente, si se ingresa su nombre y apellido, el total de Kms. recorridos hasta el momento, y los datos del viaje a contratar: precio, destino, Kms. a recorrer. Informar los datos ingresados, el monto del descuento y el monto a pagar por el cliente.

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string nya, destino;
    int kms, kmrec, total;
    float precio, desc, imp;

    cout<<"Ingrese los siguientes datos:"<<endl;
    cout<<"Nombre: ";
    getline(cin,nya);
    cout<<"Total Kms recorridos: ";
    cin>>kmrec;
    cout<<"Precio: $ ";
    cin>>precio;
    cin.get();
    cout<<"Destino: ";
    getline(cin,destino);
    cout<<"Kms a recorrer: ";
    cin>>kms;
    total=kmrec+kms;

    if(total>3500)
    {
        desc=precio*0.30;
    }else if(total>2000)
    {
        desc=precio*0.15;
    }else if(total>500)
    {
        desc=precio*0.05;
    } else desc=0;

    imp=precio-desc;

    cout<<endl; //linea en blanco
    cout<<"Cliente: "<<nya<<" Acumulados:  "<<kmrec<<endl;
    cout<<"Datos del viaje: $"<<precio<<" "<<kms<<" Kms."<<" "<<destino<<endl;
    cout<<"Descuento: $"<<desc<<" Importe: $"<<imp;
    return 0;
}
```

b. Estructura de selección múltiple

Esta estructura es una generalización de la estructura condicional y brinda la posibilidad de elegir entre más de dos alternativas de acuerdo al valor de una variable de control.
En pseudocódigo y en algunos lenguajes de programación, esta variable de control debe ser siempre del tipo numérica.

En diagrama de flujo se representa de la siguiente manera:



donde **E** es la variable de control, **numérica,** que toma todos los valores enteros positivos, acotados dentro de un rango preestablecido, generalmente de 1 a N.

A, B, ..R,. U representan una acción elemental.

En presencia de esta estructura el procesador deberá:

- observar el valor de la variable E
- decidir el camino a ejecutar de acuerdo al valor de E:
- Si E toma el valor 1, realizará la acción A
- Si E toma el valor 2, ejecutará la acción B y así sucesivamente para cada uno de los valores posibles de E que se hayan indicado.
- Si E toma valores fuera del rango 1 a N, deberá ejecutar la acción U, indicada en "otros".
- en todos los casos, ejecutada la acción asociada, deberá salir de la estructura y continuar con la acción siguiente especificada después del punto.

La estructura de decisión múltiple, mantiene las características vistas en las anteriores, de tener **un único punto de entrada y un único punto de salida.**

La sintaxis equivalente en pseudocódigo es la siguiente:

```
SEGUN E HACER
        1: A
        2: B
        . .
        . .
        n: R
      DEOTROMODO: U
FINSEGUN
```

El comienzo y el fin de la estructura están identificados por las palabras claves **SEGÚN** y **FINSEGÚN**, respectivamente. Ejecutada la acción asociada al valor de la variable **E**, el ejecutante finaliza la estructura y continúa con la acción siguiente a la palabra clave FINSEGÚN.

En C++, esta estructura es ***switch***: Esta sentencia permite efectuar una selección entre múltiples opciones en base al valor de una variable de control que permite administrar la estructura (la variable de control sólo puede ser variables de tipo **int** o **char**). Es similar a la sentencia *case* o *select* de otros lenguajes o el *según* que se emplea en pseudocódigo.

Sintaxis	Ejemplo

`switch (variable)`
`{`
`  case valor1: acción_1;`
`  break;`
`  case valor2: acción_2;`
`  break;`
`  case valor3: acción_3;`
`  break;`
`  .....`
`  default:  acción_m;`
`}`
	`switch ( m )`
	`  {`
	`  case 1: m++;`
	`  break;`
	`  case 2: m=2*m;`
	`  break;`
	`  case 3: m = m / 2;`
	`  break;`
	`  default : m = 100;`
	`  break;`
	`  }`

La acción propuesta a continuación de **default** se ejecutará si el valor de la variable de control no coincide con ninguno de los valores propuestos en la lista. La opción default es opcional; si no se indica y el valor de la expresión no aparece en la lista propuesta, ninguna acción será ejecutada.

Puede suceder que para distintos valores de la variable de control se deban ejecutar la/s misma/s acción/es, la estructura permite agrupar dichos valores indicando el camino de acciones a realizar, por única vez.

Ejemplo

```
switch ( m )
   {
   case 1:
   case 4: m++;
          break;
   case 2: m=2*m;
          break;
   case 3: m = m / 2;
          break;
   default : m = 100;
          break;
   }
```

En este ejemplo se consideró que para los valores de **m** igual a 1 ó 4, se debe ejecutar la misma acción: m++.

Ejemplo Práctico 4:
Se desea calcular el sueldo de un empleado, conociendo como datos su legajo, nombre y apellido, categoría (1, 2, 3, ó 4) y sueldo básico. Se le paga además, una bonificación que depende de la categoría. Para la categoría 1 y 3, la bonificación es del 30 % del sueldo

básico, para la categoría 2, del 20 % y para la categoría 4, del 15 %. Informar legajo, sueldo básico, bonificación y sueldo a cobrar. (En el dato categoría, sólo pueden venir los valores indicados).

```
#include <iostream>
#include <string>
using namespace std;

int main() {

    string nya;
    int leg, cat;
    float sb, bon, tot;

    cout<<"Ingrese los siguientes datos:"<<endl;
    cout<<"NRO LEGAJO: ";
    cin>>leg;
    cin.get();
    cout<<"NOMBRE: ";
    getline(cin,nya);
    cout<<"CATEGORIA (1,2,3,4): ";
    cin>>cat;
    cout<<"SUELDO BASICO: $ ";
    cin>>sb;

    switch (cat)
    {
    case 1:
    case 3: bon=sb*0.30;
            break;
    case 2: bon=sb*0.20;
            break;
    case 4: bon=sb*0.15;
            break;
    }

    tot=sb+bon;
    cout<<endl; //linea en blanco
    cout<<"LEGAJO: "<<leg<<" SUELDO BASICO: $"<<sb<<"  BONIFIC.: $"<<bon<<endl;
    cout<<"SUELDO A COBRAR: $"<<tot;
    return 0;
}
```