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

FUNDAMENTOS DE PROGRAMACIÓN

RESUMEN DE CONTENIDOS N°8

Estructuras de Control en C++

RESUMEN DE CONTENIDOS N° 8

Estructuras de Control en C++

Introducción

Lla 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.

Estas estructuras ya fueron vistas en unidades anteriores. A continuación, se planteará cada una de ellas en el lenguaje C++.

ESTRUCTURAS LÓGICAS DE CONTROL

SECUENCIA

En pseudocódigo o C++, 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

Observamos 2 tipos de estructuras de selección:

- condicional o de decisión o **if**
- selección múltiple o **switch**

If (Estructura condicional o de decisión)

La sintaxis en pseudocódigo sería:

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

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.

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 pseudocódigo:

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

En C++:

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

```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>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;
}
```

Switch (Estructura de selección múltiple)

La sintaxis equivalente en pseudocódigo es la siguiente:

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

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;
}
```

WHILE (Mientras)

La estructura **Mientras**, se representa como:

MIENTRAS condición **HACER**

A

FINMIENTRAS

Las palabras claves **MIENTRAS** y **FINMIENTRAS** identifican, respectivamente, el principio y fin de la estructura.

En C++, esta estructura es equivalente al **While**. Las acciones abarcadas por esta estructura se ejecutan repetidamente mientras la expresión lógica arroje un valor **distinto de cero** (verdadero).

Sintaxis	Ejemplo

`while (expresión lógica )`
`{`
`  acciones`
`}`
	`int a=0;`
	`while ( a<100 )`
	`{`
	`  cout << a<< "\n";`
	`  a++;`
	`}`

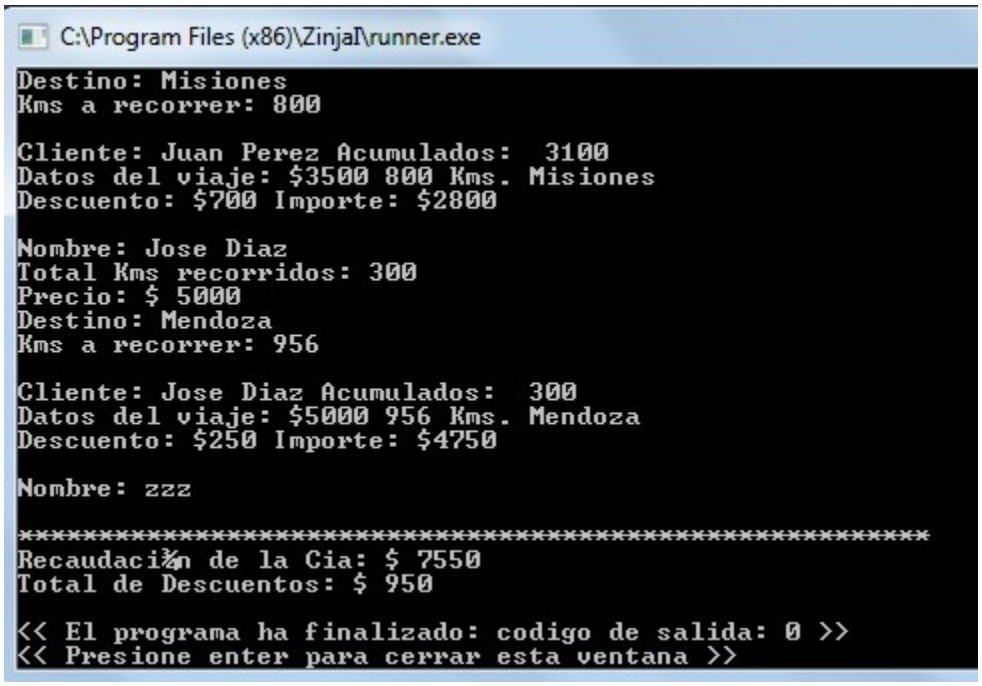
Ejemplo Práctico 5:
Una compañía de turismo ha definido una política de promoción de sus empresas para lo cual ha fijado descuentos para sus clientes en función de los viajes anteriores realizados. El valor del descuento es del 20% para aquellos clientes que, considerando los viajes realizados en el 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 la recaudación de la compañía en un día. Para ello, por cada cliente 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. El fin de datos se produce al ingresar como nombre y apellido un valor "ZZZ" .
Para cada cliente informar los datos ingresados, el monto del descuento y el monto a pagar. Informar además el total recaudado y el total de descuentos realizados por la Compañía, con leyendas alusivas.

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

int main() {
    string nya, destino;
    int kms, kmrec, total;
    float precio, desc, imp;
    float totdesc=0;
    float totrec=0;

    cout<<"Ingrese los siguientes datos:"<<endl;
    cout<<"Nombre: ";
    getline(cin,nya);
    while(nya!="zzz")
    {
        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;
        cout<<"Cliente: "<<nya<<" Acumulados:  "<<kmrec<<endl;
        cout<<"Datos del viaje: $"<<precio<<" "<<kms<<" Kms."<<" "<<destino<<endl;
        cout<<"Descuento: $"<<desc<<" Importe: $"<<imp<<endl;
        cout<<endl;
        totdesc=totdesc+desc;
        totrec=totrec+imp;
        cout<<"Nombre: ";
        cin.get();
        getline(cin,nya);
    }
    cout<<endl; //linea en blanco
    cout<<"*****************************************************"<<endl;
    cout<<"Recaudación de la Cia: $ "<<totrec<<endl;
    cout<<"Total de Descuentos: $ "<<totdesc;
    return 0;
}
```

Seguimiento:

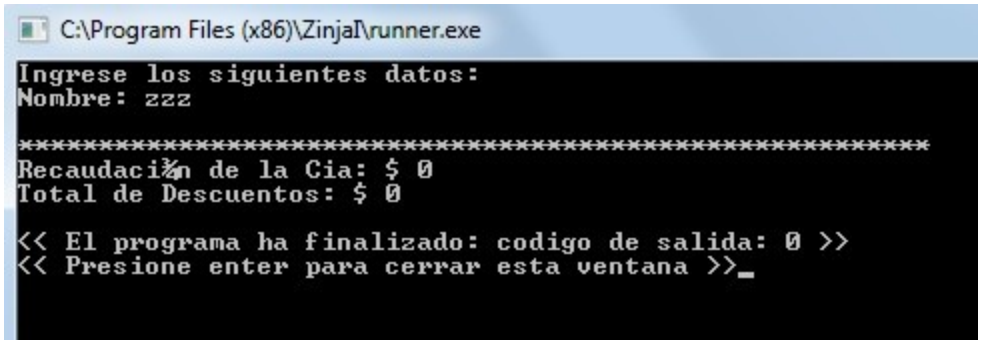
a) Datos: “Juan Pérez”, 3100, 3500, “Misiones”, 800
“José Díaz”, 300, 5000, “Mendoza”, 956
“ZZZ”

NYA	KMREC	PRECIO	KMS	DEST	TOTAL	DESC	IMP	TOTDES	TOTREC
“Juan Pérez”	3100	3500	800	“Misiones”	3900	700	2800	0	0
								700	2800
“José Díaz”	300	5000	956	“Mendoza”	1256	250	4750	950	7550
“ZZZ”									



b) Datos: "ZZZ"

NYA	KMREC	PRECIO	KMS	DEST	TOTAL	DESC	IMP	TOTDES	TOTREC
"ZZZ"								0	0



Observación: debemos notar que en este ejemplo se desconoce a priori la cantidad de veces que se repite el proceso. Sin embargo, el mismo algoritmo puede utilizarse para el caso de `ningún' cliente (ejemplo b), para un solo cliente, para 2 (ejemplo a) o para **n** clientes.

DO-WHILE (puede asemejarse a la estructura Repetir)

La estructura **Repetir**, se representa como:

REPETIR

A

HASTAOUE condición

En C++, esta estructura es equivalente al **Do-While**. Las acciones abarcadas por esta estructura se ejecutan repetidamente hasta que la expresión lógica arroje el resultado **cero** (falso).

Sintaxis	Ejemplo

`Do`
`{`
`  acciones`
`} while (expresión lógica );`
	`int b=0;`
	`do`
	`{`
	`  b++ ;`
	`  cout << b<< "\n" ;`
	`}  while ( b<100 );`

Ejemplo Práctico 6:

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 definir un algoritmo que calcule la recaudación diaria de la compañía para sus N clientes. Para ello, por cada cliente 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. Para cada cliente informar los datos ingresados, el monto del descuento y el monto a pagar. Informar además el total recaudado y el total de descuentos efectuados por la Compañía, con leyendas alusivas.

El valor N se ingresa como primer dato.

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

int main() {
    string nya, destino;
    int kms, kmrec, total;
    float precio, desc, imp;
    float totdesc=0;
    float totrec=0;
    int cont=0;
    int N;

    cout<<"Ingrese cantidad de clientes a cargar: ";
    cin>>N;
    cout<<endl;
    cout<<"Ingrese los siguientes datos:"<<endl;
    cin.get();
    do
    {
        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;
       cout<<"Cliente: "<<nya<<" Acumulados:  "<<kmrec<<endl;
       cout<<"Datos del viaje: $"<<precio<<" "<<kms<<" Kms."<<" "<<destino<<endl;
       cout<<"Descuento: $"<<desc<<" Importe: $"<<imp<<endl;
       cout<<endl;
       totdesc=totdesc+desc;
       totrec=totrec+imp;
       cont++;
       cin.get();
    }while(cont<N);
    cout<<endl;
    cout<<"************************************************************"<<endl;
    cout<<"Recaudación de la Cia: $ "<<totrec<<endl;
    cout<<"Total de Descuentos: $ "<<totdesc;
    return 0;
}
```

Seguimiento:

Datos: 2, “Juan Pérez”, 3100, 3500, “Misiones”, 800
“José Díaz”, 300, 5000, “Mendoza”, 956

NYA	KMREC	PRECIO	KMS	DEST	TOTAL	DESC	IMP	TOTDES	TOTREC	N	CONT
“Juan Pérez”	3100	3500	800	“Misiones”	3900	700	2800	0	0	2	0
								700	2800		1
“José Díaz”	300	5000	956	“Mendoza”	1256	250	4750	950	7550		2

```
Ingrese cantidad de clientes a cargar: 2

Ingrese los siguientes datos:
Nombre: Juan Perez
Total Kms recorridos: 3100
Precio: $ 3500
Destino: Misiones
Kms a recorrer: 800

Cliente: Juan Perez Acumulados:  3100
Datos del viaje: $3500 800 Kms. Misiones
Descuento: $700 Importe: $2800

Nombre: Jose Diaz
Total Kms recorridos: 300
Precio: $ 5000
Destino: Mendoza
Kms a recorrer: 956

Cliente: Jose Diaz Acumulados:  300
Datos del viaje: $5000 956 Kms. Mendoza
Descuento: $250 Importe: $4750


************************************************************
Recaudación de la Cia: $ 7550
Total de Descuentos: $ 950

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

Observación: vemos en este ejemplo que conocemos de antemano la cantidad de clientes a procesar a través de la variable N.

FOR (Para)

La estructura de repetición llamada **PARA** nos permite realizar un conjunto de acciones un número determinado de veces.

```
Para I desde VI hasta VF con Paso P Hacer
        A
FinPara
```

donde el principio y el fin de la estructura están dados por las palabra claves **PARA** y **FINPARA.**

En C++, esta estructura es equivalente al **For**. Las acciones abarcadas por esta estructura se ejecutan repetidamente hasta que la **exp2** arroje **cero** (falso); **exp1** es una expresión de inicialización y se ejecutan una única vez; **exp3** se realiza al final del grupo de acciones y generalmente se emplea para incrementar la variable que controla la estructura.

Sintaxis	Ejemplo

`for (exp1; exp2; exp3)`
`{`
`  acciones`
`}`
	`int a=0;`
	`for ( a=0 ; a<100 ; a++ )`
	`  cout << a<< "\n" ;`

Ejemplo Práctico 7:

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 definir un algoritmo que calcule la recaudación diaria de la compañía para sus N clientes. Para ello, por cada cliente 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. Para cada cliente informar los datos ingresados, el monto del descuento y el monto a pagar. Informar además el total recaudado y el total de descuentos efectuados por la Compañía, con leyendas alusivas.

El valor N se ingresa como primer dato.

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

int main() {
    string nya, destino;
    int kms, kmrec, total;
    float precio, desc, imp;
    float totdesc=0;
    float totrec=0;
    int i;
    int N;

    cout<<"Ingrese cantidad de clientes a cargar: ";
    cin>>N;
    cout<<endl;
    cout<<"Ingrese los siguientes datos:"<<endl;
    cin.get();
    for(i=0;i<N;i++)
    {
        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;
```

```
32          total=kmrec+kms;
33          if(total>3000)
34              desc=precio*0.20;
35          else desc=precio*0.05;
36          imp=precio-desc;
37          cout<<endl;
38          cout<<"Cliente: "<<nya<<" Acumulados:  "<<kmrec<<endl;
39          cout<<"Datos del viaje: $"<<precio<<" "<<kms<<" Kms."<<" "<<destino<<endl;
40          cout<<"Descuento: $"<<desc<<" Importe: $"<<imp<<endl;
41          cout<<endl;
42          totdesc=totdesc+desc;
43          totrec=totrec+imp;
44          cin.get();
45      };
46      cout<<endl;
47      cout<<"*************************************************************"<<endl;
48      cout<<"Recaudación de la Cia: $ "<<totrec<<endl;
49      cout<<"Total de Descuentos: $ "<<totdesc;
50      return 0;
51  }
52
```

Seguimiento:

Datos: 2, “Juan Pérez”, 3100, 3500, “Misiones”, 800
“José Díaz”, 300, 5000, “Mendoza”, 956

```
Ingrese cantidad de clientes a cargar: 2

Ingrese los siguientes datos:
Nombre: Juan Perez
Total Kms recorridos: 3100
Precio: $ 3500
Destino: Misiones
Kms a recorrer: 800

Cliente: Juan Perez Acumulados:  3100
Datos del viaje: $3500 800 Kms. Misiones
Descuento: $700 Importe: $2800

Nombre: Jose Diaz
Total Kms recorridos: 300
Precio: $ 5000
Destino: Mendoza
Kms a recorrer: 956

Cliente: Jose Diaz Acumulados:  300
Datos del viaje: $5000 956 Kms. Mendoza
Descuento: $250 Importe: $4750


*************************************************************
Recaudación de la Cia: $ 7550
Total de Descuentos: $ 950

<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```

El operador coma y la sentencia for

C++ permite a través del operador coma (**,**) realizar más de una instrucción donde generalmente se admite una.

Por ejemplo, en el ciclo for, la primera expresión es usada comúnmente para inicializar una variable y la tercera expresión para modificar la variable que controla la estructura. Empleando el operador coma, puede efectuarse más de una inicialización e incremento (o decremento) de las variables inicializadas.

Ejemplo

```
int i, j;
for (i=0, j=10; i < 10 ; i++, j--)
    cout << i << " " << j << endl;
```

SALTO NO CONDICIONAL O INTERRUPCIÓN (Break y Continue)

Ambas sentencias interrumpen la ejecución del grupo de acciones abarcadas por una estructura repetitiva, saltando al final de la estructura.
Luego de la interrupción, **break**, continúa con la sentencia que sigue a la iteración, abandonando la estructura de repetición; **continue** en cambio, salta al final de la estructura de repetición pero no la abandona, y permite continuar con la próxima iteración.

Ejemplo de `break`	Ejemplo de `continue`

`int a=0;`
`while (a<5)`
`{`
`  a++;`
`  if a == 4 break;`
`  cout << a;`
`}`
	`int a=0;`
	`while (a<5)`
	`{`
	`  a++;`
	`  if a == 4 continue;`
	`  cout << a;`
	`}`

Salida: 1 2 3	Salida: 1 2 3 5

ANIDAMIENTO DE ESTRUCTURAS DE CONTROL

En el diseño de algoritmos, es común la utilización de estructuras lógicas de control complejas, las que se basan en la combinación de estructuras elementales.
Por ejemplo, los **condicionales anidados**, constituyen una estructura de control compleja, dado que es una estructura de decisión dentro de otra estructura de decisión. De la misma manera, los **ciclos anidados**, pues son estructuras de iteración incluidas dentro de otras siguiendo las mismas reglas: la estructura interna debe estar totalmente incluida en la estructura externa.

A continuación, se presentan algunos ejemplos:

Condicionales anidados

En pseudicódigo	En C++

`SI condicion 1`
 	
`  ENTONCES A`
`  SI condicion 2`
 	
`  ENTONCES B`
	`if (condicion 1)`
	`{`
	`  A;`
	`  if (condicion 2)`
	`  {`
	`  B;`
	`  }`

```
                SINO

                    C; D
            FINSI

  SINO

            SI  condicion 3

                ENTONCES M

                  SINO

                    R
            FINSI
            W
FINSI
```

```
        else
             {
              C; D;
             }
}
else
  {
      if ( condicion 3)
        {
         M;
        }
        else
             {
              R;
              }
        W;
  }
```

Condicionales anidados, dentro de un ciclo iterativo

En pseudicódigo	En C++

MIENTRAS condicion 1 **HACER**
 	
SI condicion 2
ENTONCES
SI condicion 3
 	
ENTONCES B
 	
FINSI
SINO
 	
A
FINSI
FINMIENTRAS
	while (condicion 1)
	{
	if (condicion 2)
	{
	if (condicion 3)
	{
	B;
	}
	}
	else
	{
	A;
	}
	}

Ciclos iterativos Para anidados

En pseudicódigo	En C++

Para I **desde** 1 **hasta** 10 **hacer**
 	
A
Para J **desde** 1 **hasta** 5 **hacer**
 	
B;
C
Finpara
Finpara
	for (i=1 ; i<11 ; i++)
	{
	A;
	for (j=1 ; j<6 ; j++)
	{
	B;
	C;
	}
	}

La función exit()

Esta función del lenguaje C++ permite interrumpir un programa, devolviendo un valor al entorno o plataforma empleado (DOS, UNIX, LINUX).
Se halla definida en **stdlib.h**, por lo cual, si se desea utilizar esta función, deberá incluirse este archivo en la cabecera del programa. La función, devuelve el valor de su argumento: **void exit(int)**

El valor entero que se indica como argumento se retorna al proceso padre que invocó al programa. Los valores que devuelve pueden ser diferentes, pero un valor igual a cero, indica que la interrupción del programa se ha efectuado con éxito. Un valor entero distinto de cero, indica que la interrupción del programa se ha debido a un error.

Ejemplo

```
cout<<"Desea continuar operando con el programa (S/N)?";
cin >> resp ;
resp = toupper( resp );   // pasa a mayúsculas
if  (resp=='S') exit(0);
```