Fundamentos de Programación

Unidad 7: Estructuras de Control

FICH UNL
Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas

2019

Pablo Novara

CONDICIONAL: `if-else`

```
Si condición Entonces
    ...acciones por verdadero...
Sino
    ...acciones por falso...
FinSi
```

```
if (condición) {
    ...acciones por verdadero...
} else {
    ...acciones por falso...
}
```

- las condiciones siempre van entre paréntesis en las estructuras de control
- el anidamiento se delimita con llaves

CONDICIONAL: `if-else`

```
Si condición Entonces
    ...acciones por verdadero...


FinSi
```

```
if (condición) {
    ...acciones por verdadero...


}
```

✅ *al igual que en pseudocódigo, se puede omitir la rama del falso*

EJEMPLOS `if-else`

1. Escriba un programa para encontrar la raíces (reales o complejas) de una ecuación cuadrática.

EJEMPLO `if-else`

```
int main() {
    float a, b, c;
    cin >> a >> b >> c;
    float d = b*b-4*a*c;
    if (d>=0) { // raices reales
        float r1 = (-b+sqrt(d))/(2*a),
              r2 = (-b-sqrt(d))/(2*a);
        cout << r1 << endl << r2 << endl;
    } else { // raices complejas
        float pr = -b/2*a;
        float pi = sqrt(-d)/2*a;
        cout << pr << "+" << pi << "i" << endl;
        cout << pr << "-" << pi << "i" << endl;
    }

}
```

ÁMBITO DE VALIDEZ DE UNA VARIABLE (SCOPE)

```
int main() {
   float a, b, c;
   cin >> a >> b >> c;
   float d = b*b-4*a*c;
   if (d>=0) { // raices reales
      float r1 = (-b+sqrt(d))/(2*a),
            r2 = (-b-sqrt(d))/(2*a);
      cout << r1 << endl << r2 << endl;
   } else { // raices complejas
      float pr = -b/2*a;
      float pi = sqrt(-d)/2*a;
      cout << pr << "+" << pi << "i" << endl;
      cout << pr << "-" << pi << "i" << endl;
   }
   // r1,r2,pr,pi ya no son válidas aquí
}
```

✅ *declarar las variables lo más "tarde" posible*

EJEMPLOS

1. Escriba un programa para encontrar la raíces (reales o complejas) de una ecuación cuadrática.

2. Escriba un programa para determinar si un triángulo es de tipo rectángulo a partir de las longitudes de sus tres lados.

advertencia: no comparar resultados reales con ==, no suelen ser exactos

USO DE LLAVES PARA DELIMITAR BLOQUES

Las llaves agrupan instrucciones y crean un *scope*:

```
if (condición) {
    acciones por verdadero
}
acciones fuera del if
```

Cuando no hay llaves, la estructura de control abarca sólo la primer instrucción o estructura:

```
if (condición)
    una acción por verdadero;
acciones fuera del if
```

eviten omitir las llaves en las primeras prácticas con C++

OMISIÓN DE LLAVES

¿Cuál de estas dos indentaciones es la correcta?

```
if (condición 1)
    if (condicion 2)
        acción por verdadero
else
    acción por falso 1
```

```
if (condición 1)
    if (condicion 2)
        acción por verdadero
    else
        acción por falso 2
```

Respuesta correcta: **No importa!!!**

Eviten el problema, usen llaves para clarificar

OMISIÓN DE LLAVES

- *Las llaves eliminan la ambigüedad (ahora hay un `if` "afuera" y otro "adentro")*

```
if (condición 1) {
    if (condicion 2)
        acción por verdadero
} else
    acción por falso 1
```

```
if (condición 1) {
    if (condicion 2)
        acción por verdadero
    else
        acción por falso 2
}
```

ITERATIVA: `while`

```
Mientras condición Hacer
    ...acciones a repetir...
FinMientras
```

```
while (condición) {
    ...acciones a repetir...
}
```

EJEMPLOS

3. Modificar el ejemplo 2 (triángulo) para que valide las entradas (los lados nunca pueden ser negativos).

4. Se leen las temperaturas medias mensuales de un año y se desea determinar cuál fue la temp. media anual.

ITERATIVA: do-while

```
Repetir
    ...acciones a repetir...
Hasta que condición
```

```
do {
    ...acciones a repetir...
} while (condición_opuesta);
```

- do-while repite "mientras" se cumple la condición
- lleva punto y coma al final

EJEMPLOS

3. Modificar el ejemplo 2 (triángulo) para que valide las entradas (los lados nunca pueden ser negativos).

4. Se leen las temperaturas medias mensuales de un año y se desea determinar cuál fue la temp. media anual.

ITERATIVA: for

```
Para i<-1 Hasta N Con Paso P Hacer
    ...acciones a repetir...
FinPara
```

```
for ( i=1 ; i<=N ; i+=P ) {
    ...acciones a repetir...
}
```

ITERATIVA: `for`

```
Para i<-1 Hasta N Con Paso P Hacer
    ...acciones a repetir...
FinPara
```

```
for ( i=1 ; i<=N ; i+=P ) {
```

inicialización del contador

condición tipo mientras

incremento del contador

```
    ...acciones a repetir...
}
```

ITERATIVA: `for`

```
int i;
for ( i=1 ; i<=N ; i+=P ) {
    ...acciones a repetir...
}
```

Se puede definir en el **`for`**

a la variable que hace de contador :

```
for ( int i=1 ; i<=N ; i+=P ) {
    ...acciones a repetir...
}
```

el scope de la variable es el "interior" del for

es mejor definir al contador en el for

ITERATIVA: `for`

```
for ( int i=1 ; i<=N ; i+=1 ) {
    ...acciones a repetir...
}
```

```
for ( int i=1 ; i<=N ; ++i ) {
    ...acciones a repetir...
}
```

✅ *usualmente para el incremento usamos el operador ++*

```
for ( int i=0 ; i<N ; ++i ) {
    ...acciones a repetir...
}
```

✅ *los que programan en C/C++ tienen la "extraña" costumbre de contar desde 0*

EJEMPLOS

5. Promediar N números reales.

```
float sum=0;
int N;
cin >> N;
for ( int i=0; i<N; ++i ) {
    int aux;
    cin >> aux;
    sum += aux;
}
cout << sum/N << endl;
```

`break` Y `continue`

Se utilizan dentro de estructuras iterativas:

- **`break`** sale inmediatamente de una estructura iterativa

- **`continue`** saltea lo que queda de la iteración actual

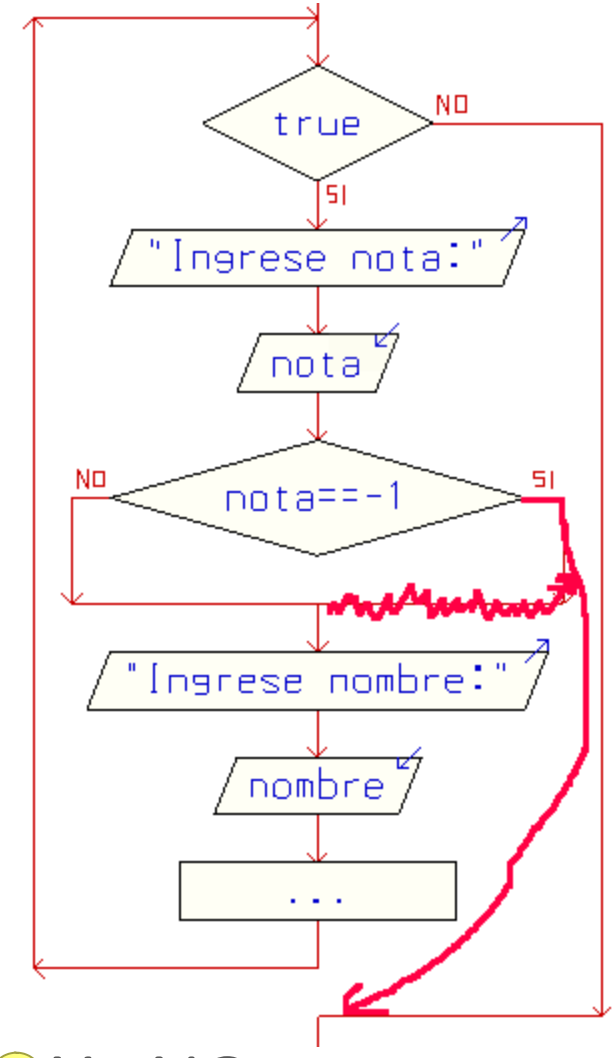
USO DE `break`

Ejemplo: se leen notas y nombres hasta que la nota sea -1...

```
cin >> nota;
while (nota!=-1) {
    cin >> nombre;
    ...procesar datos...
    cin >> nota;
}
```

```
while (true) {
    cin>>nota;
    if (nota==-1) break;
    cin>>nombre;
    ...procesar datos...
}
```

USO DE break



- *Ya NO es programación estructurada*

- *Úsese con responsabilidad*

USO DE `break`

Ej: encontrar la 1ra ocurrencia de x en un vector v:

```
int pos = -1; // -1: todavía no lo encontré
int i=0;
while (i<N && pos==-1)  {
    if (v[i]==x)
        pos=i;
    ++i;
}
```

```
int pos=-1;
for (int i=0;i<N;++i) {
    if (v[i]==x) {
        pos=i;
        break;
    }
}
```

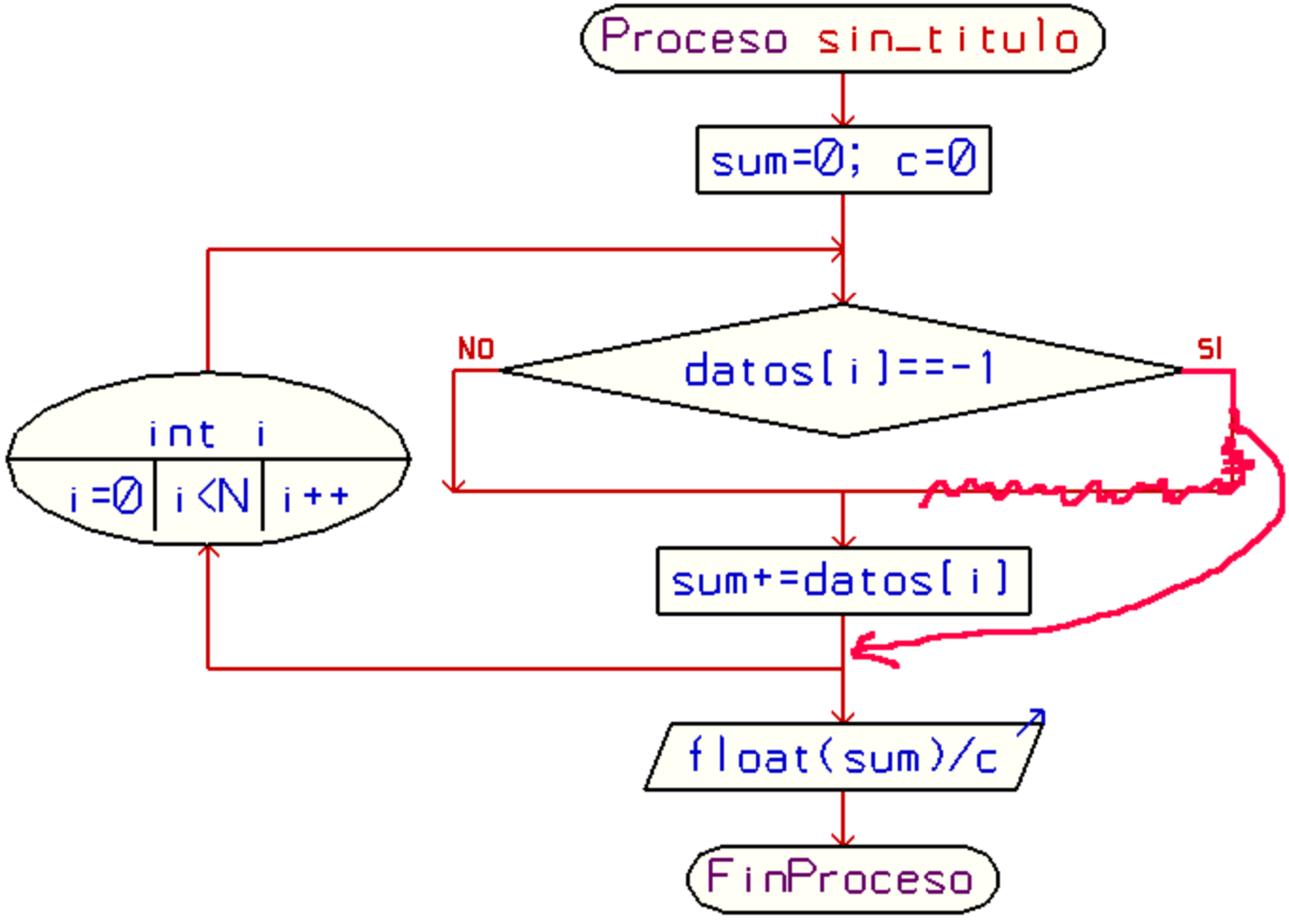
USO DE `continue`

Ejemplo: sumar y contar los elementos de un arreglo ignorando las posiciones que contienen -1:

```
int sum=0, c=0;
for (int i=0;i<N;++i) {
    if (datos[i]!=-1) {
        sum+=datos[i];
        ++c;
    }
}
```

```
int sum=0, c=0;
for (int i=0;i<N;++i) {
    if (datos[i]==-1) continue;
    sum+=datos[i];
    ++c;
}
```

USO DE `continue`



idem slide de break

SELECCIÓN MÚLTIPLE: `switch`

```
Segun expresion_numerica Hacer
    valor1:
        ...acciones para valor1...
    valor2:
        ...acciones para valor2...
    valor3:
        ...acciones para valor3...
    De Otro Modo:
        ...acciones para otro caso...
FinSegun
```

```
switch (expresion_numerica) {
    case valor1:
        ...acciones para valor1...
        break;
    case valor2:
        ...acciones para valor2...

        break;
    case valor3:
        ...acciones para valor3...
        break;
    default:
        ...acciones para otro caso...
}
```

SELECCIÓN MÚLTIPLE: `switch`

```
switch (expresion_numerica) {
    case valor1:
        ...acciones para valor1...
        break;
    case valor2:
        ...acciones para valor2...
        break;
    case valor3:
        ...acciones para valor3...
        break;
    default:
        ...acciones para otro caso...
}
```

- cada opción lleva un ***break*** al final
- ¿qué pasa si se omite el ***break***?
- ¿para qué puede servir eso?