ESTRUCTURAS DE PROGRAMACION

- **SECUENCIA**: indica el orden de ejecución de las instrucciones. Una por vez y de arriba hacia abajo.

- **DECISIÓN:** a partir del análisis de una proposición lógica se decide por dónde continúa la ejecución del programa (verdadero o falso).
  De acuerdo al caso puede usarse una **DECISION SIMPLE** o una **DECISIÓN MÚLTIPLE**

- **CICLOS:** pemiten que un programa ejecute de manera repetitiva un conjunto de instrucciones**. De acuerdo al caso puede ser CICLO EXACTO (for) o CICLO INEXACTO (while).**

ESTRUCTURAS DE REPETICION (CICLOS)

- **Ciclo exacto**: se conoce la cantidad de veces que se necesita repetir las instrucciones. En general es implementado por los lenguajes de programación como ciclo **for.**

- **Ciclo inexacto**: no se conoce la cantidad de veces que se necesita repetir las instrucciones. En general es implementado por los lenguajes de programación como ciclo **while.**

ESTRUCTURAS DE REPETICION (CICLOS)

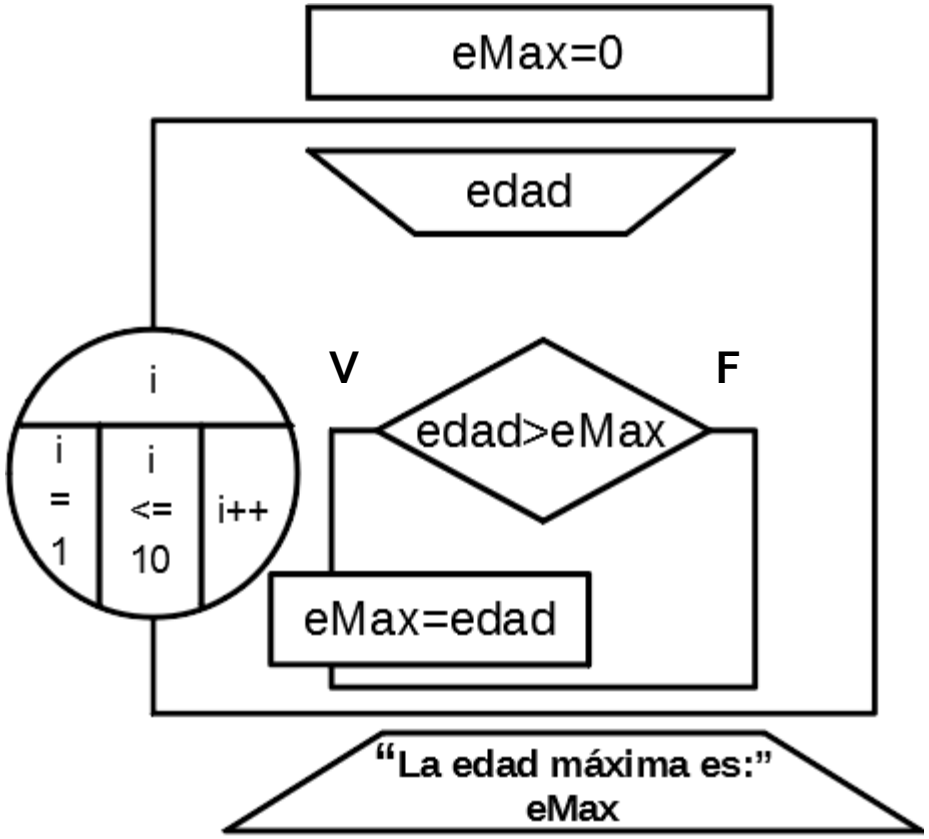
- **Ciclo exacto**: se conoce la cantidad de veces que se quiere repetir las instrucciones

- **Ejemplo**: dadas las edades de 10 alumnos de un curso, calcular e informar la edad máxima

ESTRUCTURAS DE REPETICION

Ejemplo: dadas las edades de 10 alumnos de un curso, calcular e informar la edad máxima

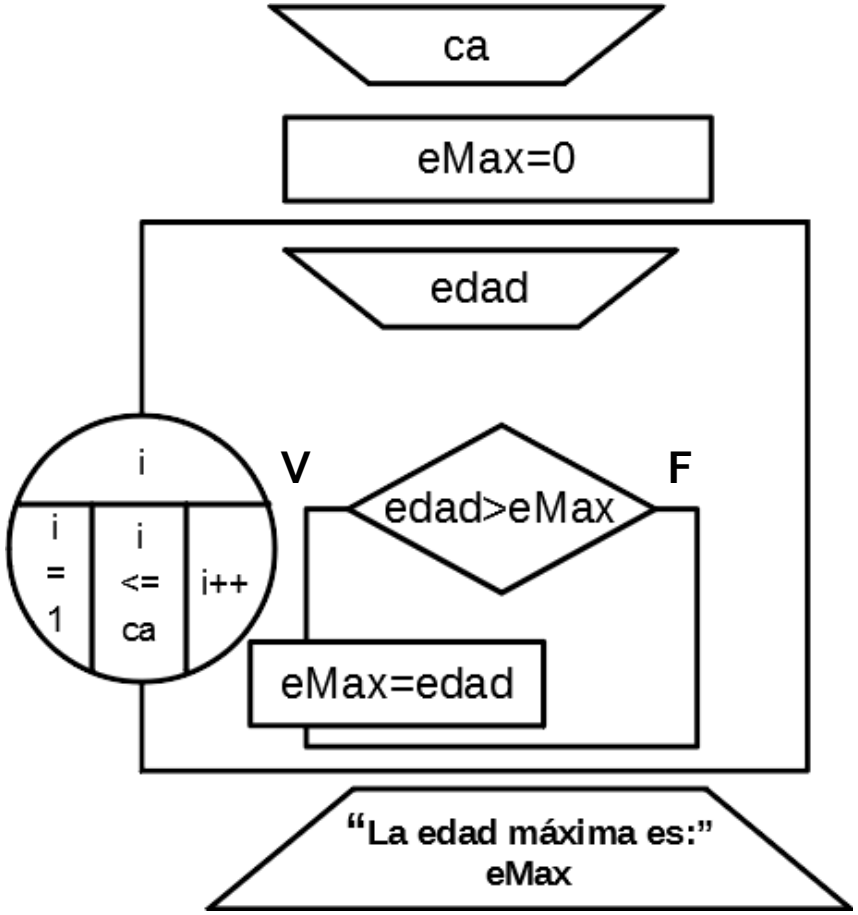
```
int main(){
    int i, edad, eMax=0;
    for(i=1;i<=10;i++){
        cout<<“INGRESE LA EDAD: “;
        cin>>edad;
        if(edad>eMax) eMax=edad;
    }
    cout<<“EDAD MAXIMA: ”<<eMax<<endl;
    return 0;
}
```



ESTRUCTURAS DE REPETICION

Ejemplo: dadas las edades de los alumnos de un curso, calcular e informar la edad máxima.
La cantidad de alumnos del curso se ingresa al inicio del programa

```
int main(){
    int i, edad, ca, eMax=0;
    cout<<“INGRESE LA CANTIDAD DE ALUMNOS: ”;
    cin>>ca;
    for(i=1;i<=ca;i++){
        cout<<“INGRESE LA EDAD: “;
        cin>>edad;
        if(edad>eMax) eMax=edad;
    }
    cout<<“EDAD MAXIMA: ”<<eMax<<endl;
    return 0;
}
```



ESTRUCTURAS DE REPETICION

Ejemplo: dadas las edades de los alumnos de un curso, calcular e informar la edad máxima.

No se conoce la cantidad de alumnos del curso.

No se puede resolver con un ciclo exacto, por lo que debe utilizarse un ciclo inexacto, o ciclo WHILE .

ESTRUCTURAS DE REPETICION

El ciclo inexacto o while permite la ejecución de un conjunto de instrucciones mientras la condición que se evalúa en cada ciclo sea verdadera.

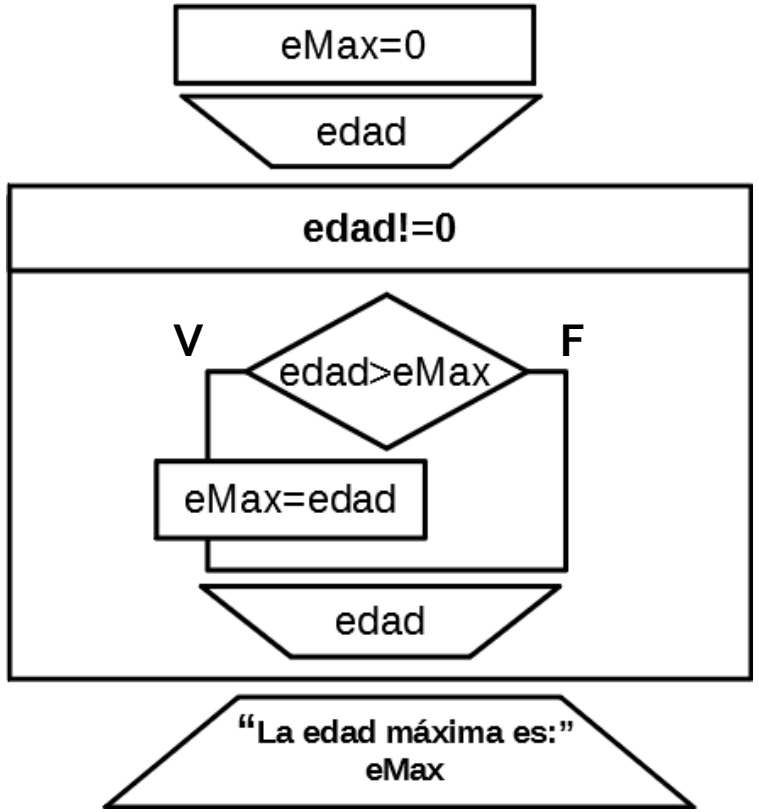
En C tiene la siguientes sintáxis:

```
while(proposición lógica a evaluar){
///instrucciones a repetir entre las llaves
}
```

ESTRUCTURAS DE REPETICION

Ejemplo: dadas las edades de los alumnos de un curso, calcular e informar la edad máxima.
Como no se conoce la cantidad de alumnos del curso debe establecerse una condición de fin. Puede ser en este caso que el programa termine cuando se ingresa un valor de edad igual a cero, ya que el cero no es un valor posible de edad.

```
int main(){
    int edad, eMax=0;
    cout<<"EDAD (0 PARA FINALIZAR): ";
    cin>>edad;
    while(edad!=0){
        if(edad>max) eMax=edad;
        cout<<"EDAD (0 PARA FINALIZAR): ";
        cin>>edad;
    }
    cout<<"EDAD MAXIMA: "<<edad<<endl;
    return 0;
}
```



ESTRUCTURAS DE REPETICION

Tanto en el ciclo for como en el while la/s variable/s que lo controlan tienen que tener valores consistentes para garantizar que el programa se ejecute correctamente.

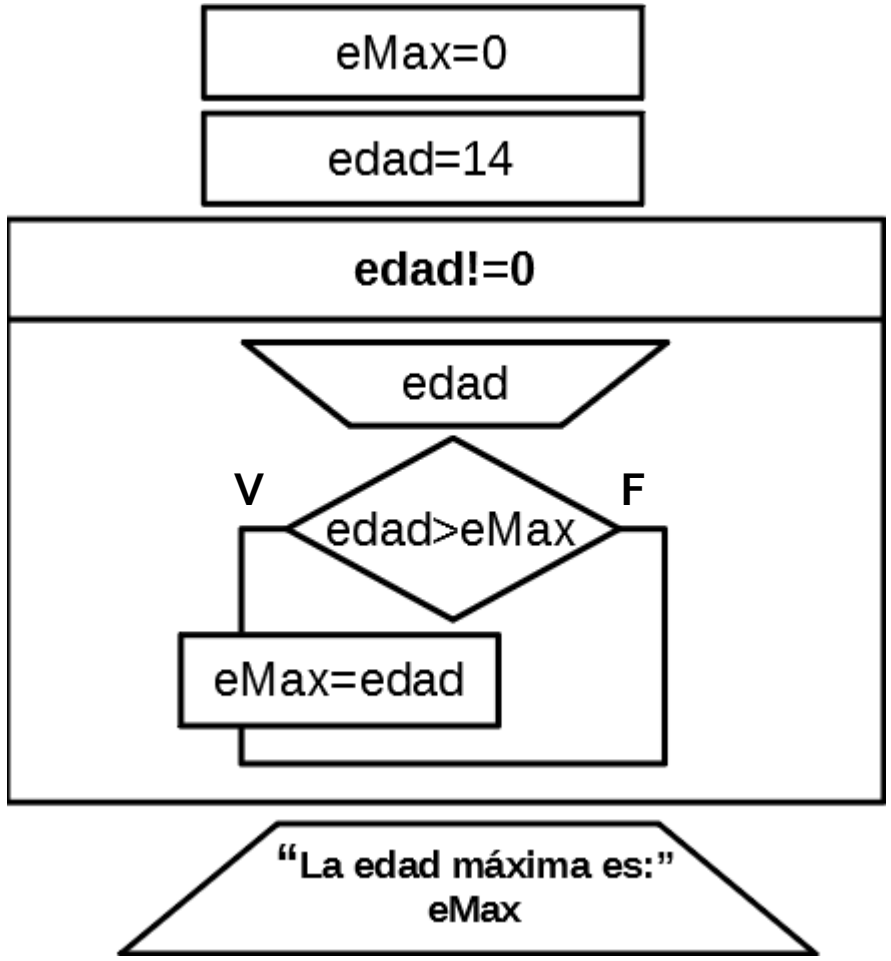
Por ejemplo en el código visto, si no se hace el ingreso previo de la edad el programa podría no pedir ingresos y dar un resultado absurdo

```
int main(){
    int edad, eMax=0;
    while(edad!=0){
        cout<<“INGRESE LA EDAD (0 PARA FINALIZAR): ”;
        cin>>edad;
        if(edad>max) eMax=edad;
        }
    cout<<“EDAD MAXIMA: ”<<edad<<endl;
    return 0;
}
```

ESTRUCTURAS DE REPETICION

Para este caso, se podría asignar un valor arbitrario a la variable edad para que el programa ingrese al ciclo

```
int main(){
    int edad, eMax=0;
    edad=14;///cualquier valor distinto de 0
    while(edad!=0){
        cout<<“EDAD (0 PARA FINALIZAR): ”;
        cin>>edad;
        if(edad>max) eMax=edad;
        }
    cout<<“EDAD MAXIMA: ”<<edad<<endl;
    return 0;
}
```



ESTRUCTURAS DE REPETICION

- En un ciclo inexacto, la condición de corte del ciclo puede obtenerse:

1. Del valor de una de las variables que se ingresa.

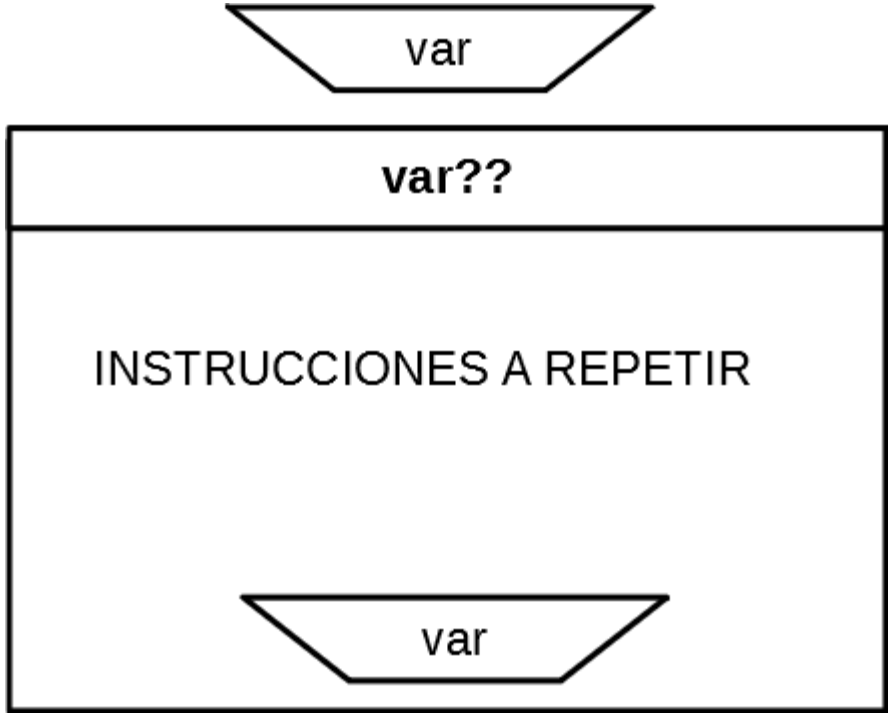
2. Del resultado del análisis de los datos que se están ingresando o procesando

ESTRUCTURAS DE REPETICION

- En un ciclo inexacto, la condición de corte del ciclo puede obtenerse:

1. Del valor de una de las variables que se ingresa. Ejemplo analizado. La forma general es:

```
cin>>var;
while(var??){
   ///instrucciones a
   ///repetir
   cin>>var;
   }
```

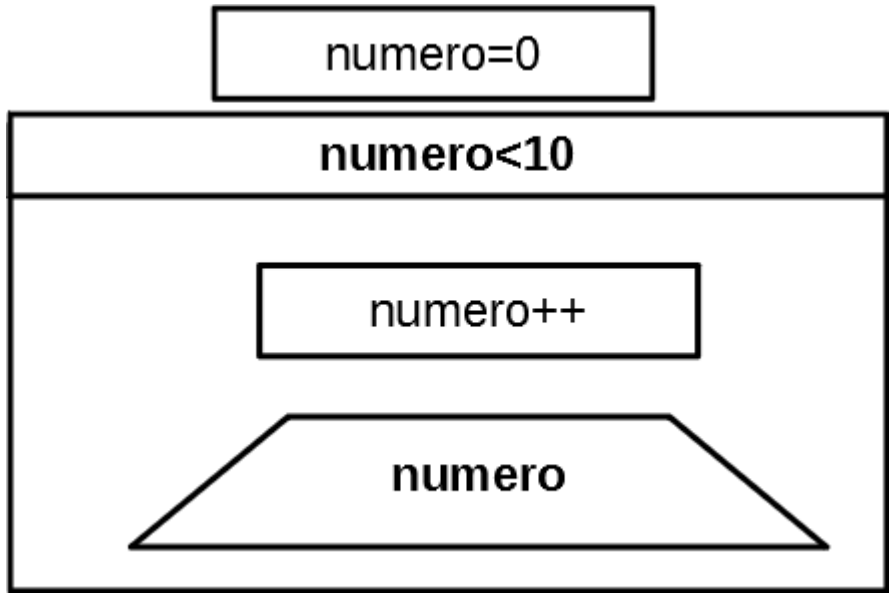


ESTRUCTURAS DE REPETICION

2. Del resultado del análisis de los datos que se están ingresando o procesando.

Ejemplo: hacer un programa muestre los números entre el 1 y el 10

```
int main(){
    int numero=0;
    while(numero<10){
        numero++;
        cout<<numero<<endl;
        }
    return 0;
}
```



ESTRUCTURAS DE REPETICION

2. Del resultado del análisis de los datos que se están ingresando o procesando.

Ejemplo: hacer un programa sume los números que se ingresan y muestre el resultado. Debe terminar cuando se ingrese por segunda vez un cero

```
int main(){
    int cantCeros=0, n, suma=0;
    while(cantCeros!=2){
        cout<<“INGRESE UN NUMERO: ”;
        cin>>n;
        suma+=n;
        if(n==0) cantCeros++;
        }
    cout<<“SUMA: ”<<suma<<endl;
    return 0;
}
```

ESTRUCTURAS DE REPETICION

2. Del resultado del análisis de los datos que se están ingresando o procesando.

Ejemplo: igual al anterior, pero utilizando una bandera en la condición del ciclo

```
int main(){
    int cantCeros=0, n, suma=0;
    bool seguir=true;
    while(seguir){///equivalente a seguir==true
        cout<<“INGRESE UN NUMERO: ”;
        cin>>n;
        suma+=n;
        if(n==0) cantCeros++;
        if(cantCeros==2) seguir=false;
        }
    cout<<“SUMA: ”<<suma<<endl;
    return 0;
}
```