UNIDAD II

Repaso de conceptos de programación en C/C++

Algoritmos

Un algoritmo es una serie de pasos finitos y ordenados para resolver un problema.

Algoritmos

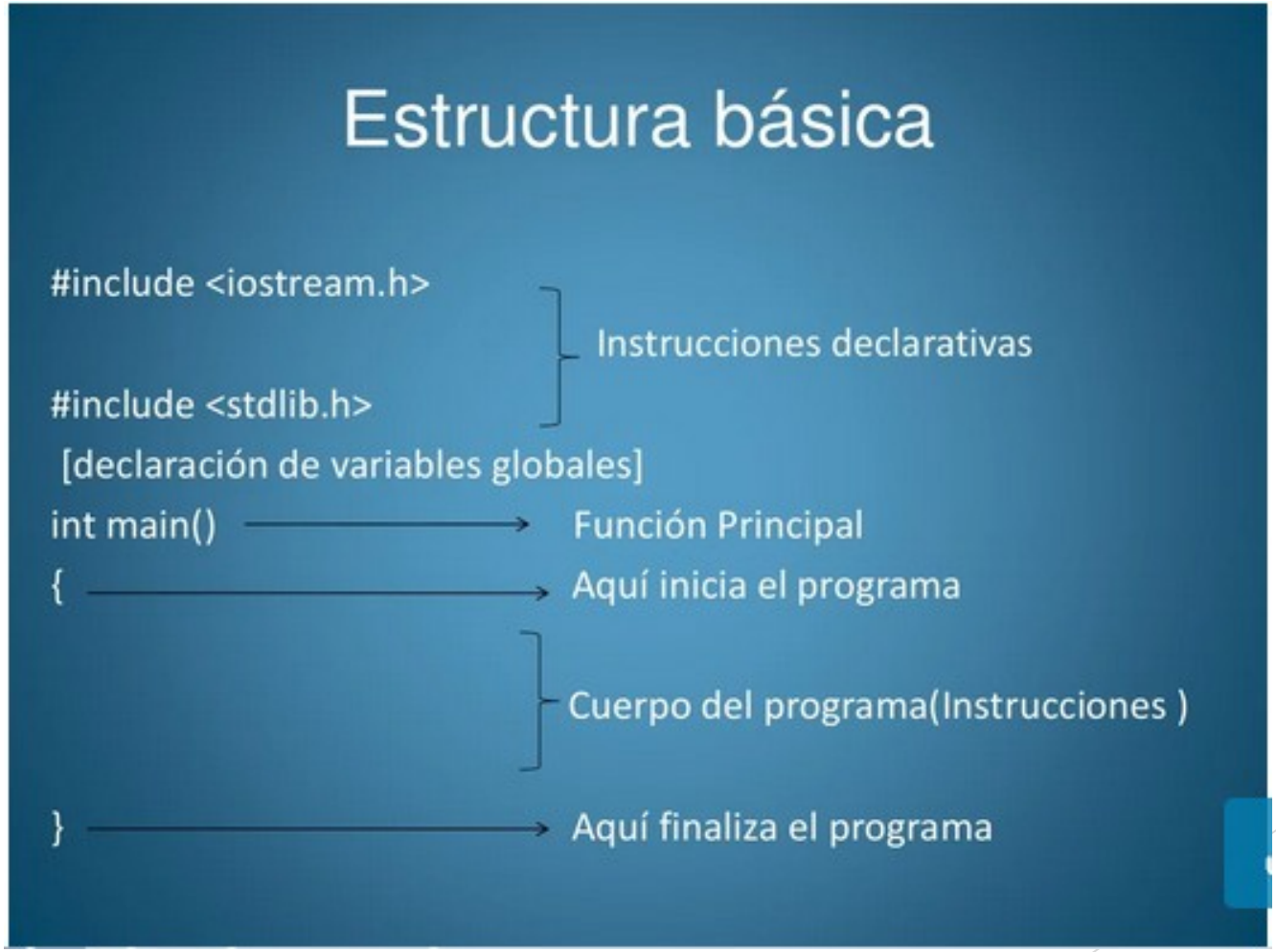
Formas de escribir un algoritmo

1 - Representarlo mediante un diagrama de flujo.

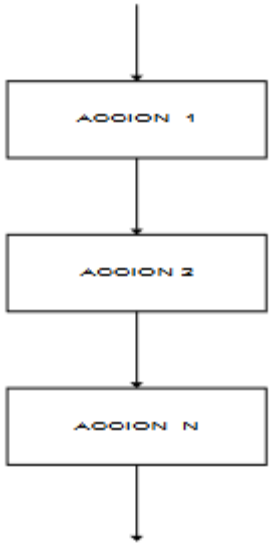
2 - Escribirlo en pseudocódigo.

3 - Escribirlo directamente en un lenguaje de programación.

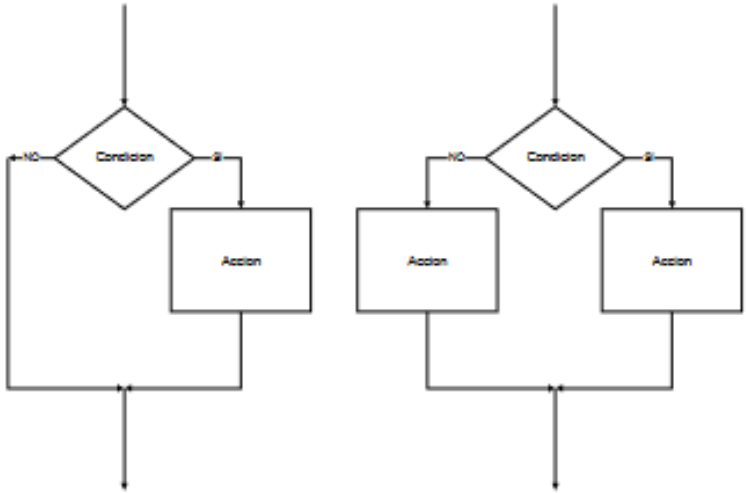
Programa C++


Estructura básica
#include <iostream.h>
#include <stdlib.h>
Instrucciones declarativas
[declaración de variables globales]
int main()
Función Principal
{
Aquí inicia el programa
Cuerpo del programa(Instrucciones)
}
Aquí finaliza el programa

Estructuras de programación

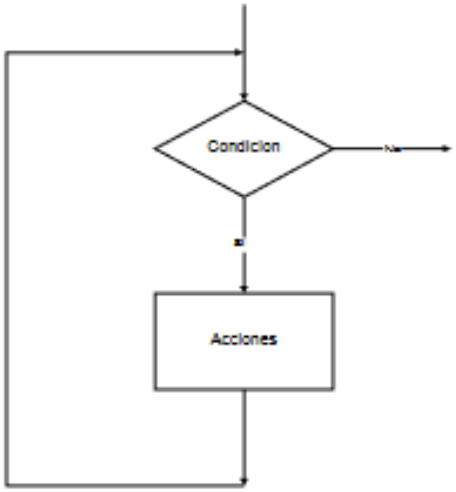


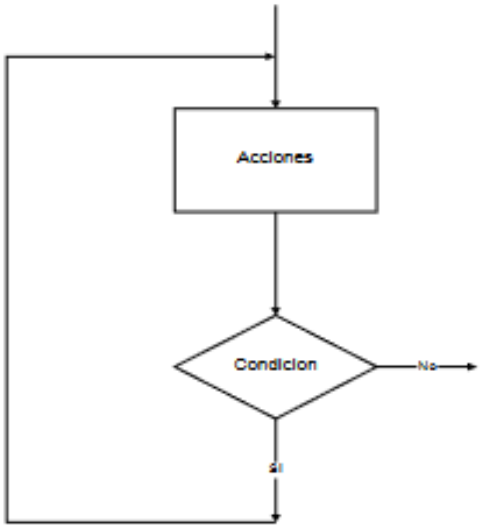
Secuencial



Condicional

Estructuras de programación





Ciclo o Repetitivas

Estructuras de programación C++

Secuencial

```
#include <iostream>

using namespace std;

int main()
{
    int num1, num2, producto;
    cout <<"Ingrese num1:";
    cin >>num1;
    cout <<"Ingrese num2:";
    cin >>num2;
    producto = num1 * num2;
    cout <<"\n";
    cout <<"El producto de los dos valores es:";
    cout <<producto;
    return 0;
}
```

Estructuras de programación C++

Condicional Simple

```
#include<iostream>

using namespace std;

int main()
{
    float cantidad;
    cout <<"Ingrese el sueldo:";
    cin >>cantidad;
    if (cantidad>1000)
    {
        cout <<"Cantidad supera el límite";
    }
    return 0;
}
```

Estructuras de programación C++

Condicional Doble

```
#include<iostream>

using namespace std;

int main()
{
    int num1, num2;
    cout <<"Ingrese num1:";
    cin >>num1;
    cout <<"Ingrese num2:";
    cin >>num2;
    if (num1 > num2)
    {
        cout <<num1;
    }
    else
    {
        cout <<num2;
    }
    return 0;
}
```

Estructuras de programación C++

Condicional Múltiple

```
using namespace std;

int main()
{
    int num1;
    cout <<"Ingrese num1:";
    cin >>num1;
    switch (num1)
    {
        case 1:
            cout<<"Valor 1";
            break;
        case 2:
            cout<<"Valor 2";
            break;
        default:
            cout<<"Otros";
            break;
    }
    return 0;
}
```

Estructuras de programación C++

Repetitiva - While

```
#include<iostream>

using namespace std;

int main()
{
    int x;
    x = 1;
    while (x <= 10)
    {
        cout <<x<<endl;
        x = x + 1;
    }
    return 0;
}
```

Estructuras de programación C++

Repetitiva – Do While

```
#include<iostream>

using namespace std;

int main()
{
    int x;
    x = 1;
    do
    {
        cout <<x<<endl;
        x = x + 1;
    }while (x<=10);
    return 0;
}
```

Estructuras de programación C++

Repetitiva - For

```
#include<iostream>

using namespace std;

int main()
{
    int f;
    for(f=1; f <= 10 ;f++)
    {
        cout <<f << endl;
    }
    return 0;
}
```

Operadores

Definición

Un operador es un elemento de programa que se aplica a uno o varios operandos en una expresión o instrucción.

Pueden ser unarios, binarios o ternarios.

Se clasifican en:
Aritméticos, Relacionales, de Asignación, Lógicos, de Dirección y de manejo de Bits.

Operadores

Asignación

Operador	Acción	Ejemplo	Resultado
=	Asignación Básica	X = 6	X vale 6
*=	Asigna Producto	X *= 5	X vale 30
/=	Asigna División	X /= 2	X vale 3
+=	Asigna Suma	X += 4	X vale 10
-=	Asigna Resta	X -= 1	X vale 5
%=	Asigna Modulo	X %= 5	X vale 1
<<=	Asigna Desplazamiento Izquierda	X <<= 1	X vale 12
>>=	Asigna Desplazamiento Derecha	X >>= 1	X vale 3
&=	Asigna AND entre Bits	X &= 1	X vale 0
^=	Asigna XOR entre Bits	X ^= 1	X vale 7
|=	Asigna OR entre Bits	X |= 1	X vale 7

Operadores

Aritméticos

Operador	Acción	Ejemplo	Resultado
-	Resta	X = 5 - 3	X vale 2
+	Suma	X = 5 + 3	X vale 8
*	Multiplicación	X = 2 * 3	X vale 6
/	División	X = 6 / 3	X vale 2
%	Módulo	X = 5 % 2	X vale 1
--	Decremento	X = 1; X--	X vale 0
++	Incremento	X = 1; X++	X vale 2

Operadores

Relacionales

Operador	Relación	Ejemplo	Resultado
<	Menor	X = 5; Y = 3; if(x < y) x+1;	X vale 5 Y vale 3
>	Mayor	X = 5; Y = 3; if(x > y) x+1;	X vale 6 Y vale 3
<=	Menor o igual	X = 2; Y = 3; if(x <= y) x+1;	X vale 3 Y vale 3
>=	Mayor o igual	X = 5; Y = 3; if(x >= y) x+1;	X vale 6 Y vale 3
==	Igual	X = 5; Y = 5; if(x == y) x+1;	X vale 6 Y vale 5
¡=	Diferente	X = 5; Y = 3; if(x ¡= y) y+1;	X vale 5 Y vale 4

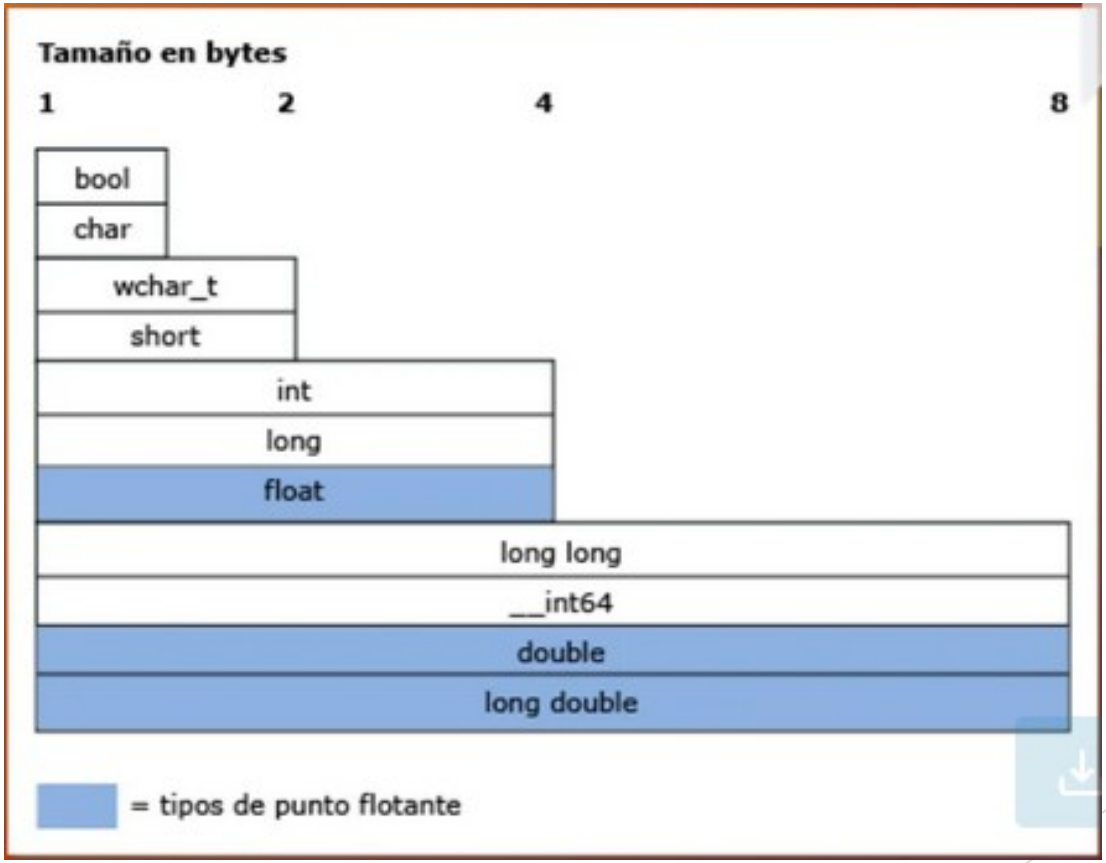
Operadores

Lógicos

Operador	Acción	Ejemplo	Resultado
&&	AND Lógico	A && B	Si ambos son verdaderos se obtiene verdadero(true)
||	OR Lógico	A || B	Verdadero si alguno es verdader
!	Negación Lógica	!A	Negación de a

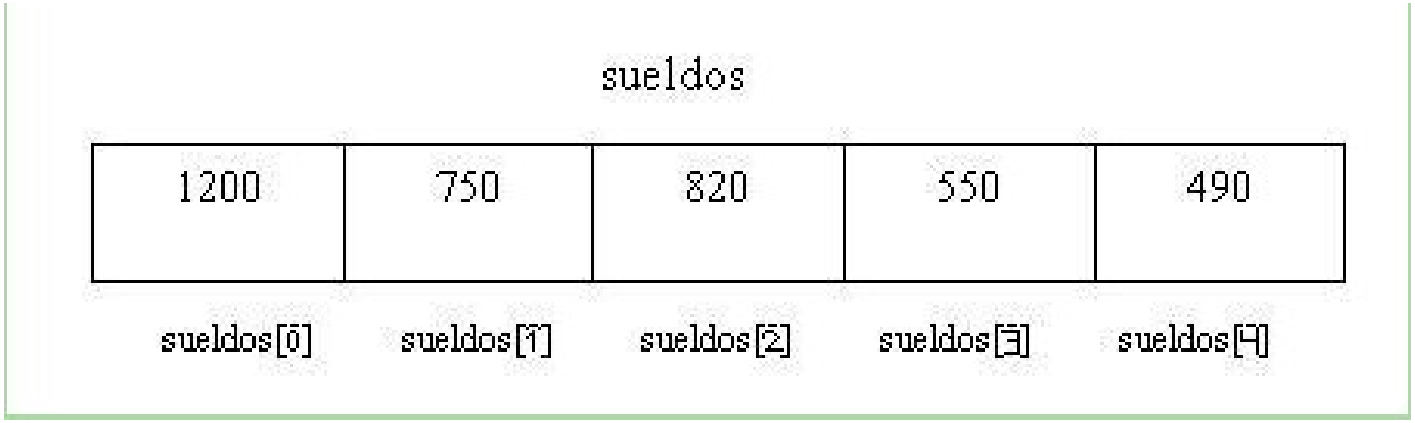
Tipos de Datos

Determinan el dato que va a contener una variable



Arreglos

Permiten almacenar un conjunto de datos bajo una misma denominación.



Arreglos

```
#include<iostream>

using namespace std;

int main()
{
    int f;
    int valores[5];
    for(f=0; f < 5 ; f++)
    {
        cin >> valores[f];
    }

    int i =0;
    while (i<5)
    {
        cout<<valores[i]<<"--";
        i++;
    }
    return 0;
}
```

Estructuras

Se pueden entender como un tipo de datos compuesto.
Permiten agrupar datos de distinto tipo

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

int main()
{
    struct
    {
        string nombre;
        int    edad;
        float saldo;
    } cliente;

    cliente.nombre = "Pedro";
    cliente.edad = 50;
    cliente.saldo = 3500;
    cout << "El saldo es " << cliente.saldo;

    return 0;
}
```

Funciones

Una función está formada por un conjunto de sentencias que realizan una determinada tarea y que podemos invocar mediante un nombre.

```
 [tipo_devuelto] nombre_funcion
 ([tipo parametro1][, tipo parametro2][, ....])
{
    // instrucciones
   [return valor;]
}
```

Funciones

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

// Declaración Función
int suma(int x, int y);

int main()
{
    int a, b;
    a = 10;
    b = 20;
    cout<<suma(a,b);

    return 0;
}

// Definición Función
int suma(int x, int y)
{
        int z;
        z = x + y;
        return z;
}
```

Funciones

```
#include<iostream>
using namespace std;
void f1(int z);
void f2(int &z);

int main()
{
    int numero = 10;
    cout << "Valor de la variable numero:" << numero << "\n";
    f1(numero);
    cout << "Valor de la variable numero:" << numero << "\n";
    f2(numero);
    cout << "Valor de la variable numero:" << numero << "\n";
    return 0;
}

void f1(int z)
{
    z = 0;
}
void f2(int &z)
{
    z = 0;
}
```