C++

Deitel & Deitel: Cómo programar en C y C++

1. Flujo de entrada/salida en C++

```
cout << "Ingrese un valor: ";

cin >> valor;

cout << "El valor ingresado es " << valor << '\n';
```

2. Ejemplo de un programa C++

```
1   // Flujo simple de entrada/salida
2   #include <iostream>
3
4   using namespace std;
5
6   int main()
7   {
8       cout << "Ingrese su edad: ";
9       int edad;
10      cin >> edad;
11
12      cout << "Ingrese la edad de su amigo: ";
13      int edadAmigo;
14      cin >> edadAmigo;
15
16      if (edad > edadAmigo)
17          cout << "Usted es el mayor\n";
18      else
19          if (edad < edadAmigo)
20              cout << "Usted es el más joven\n";
21          else
22              cout << "Ambos son de la misma edad\n";
23
24      return 0;
25  }
```

3. Declaraciones en C++

Las declaraciones pueden ser colocadas en cualquier parte del programa, siempre y cuando, anteceden el uso de lo que se está declarando.

```
cout << "Ingrese dos enteros: ";

int x, y;

cin >> x >> y;

cout << "La suma de " << x "y de " << y << "es " << x + y << "\n";
```

También pueden ser declaradas dentro de la estructura del for:

```
for (int i = 0; i <= 5; i++)
    count << i << '\n';
```

3. Declaraciones en C++

```
1   #include <iostream>
2
3   using namespace std;
4
5   int main()
6   {
7       cout << "Ingrese dos enteros: ";
8       int x, y;
9       cin >> x >> y;
10
11      cout << "La suma de " << x << " y de " << y << " es " << x + y << "\n";
12
13      return 0;
14  }
```

4. Crear nuevos tipos de datos en C++

C++ proporciona la capacidad de crear tipos definidos por el usuario mediante el uso de las palabras reservadas `enum`, `struct`, `union` y `class`.

Por ejemplo, las declaraciones:

```
enum Boleano (FALSE, TRUE);
```

```
union Numero {
    int i;
    float f;
};
```

```
struct Nombre {
    char npila[10];
    char apellidos[20];
};
```

4. Crear nuevos tipos de datos en C++

crean tres tipos de datos, definidos por el usuario, con los nombres de etiqueta `Boleano`, `Nombre` y `Numero`.

Los nombres de etiqueta pueden ser usados para declarar variables, como sigue;

```
Boleano hecho = FALSE;

Nombre estudiante;

Numero x;
```

Al igual que en C, las enumeraciones son valuadas iniciándose en cero y cada elemento subsecuente se incrementa en uno.

Por lo tanto, la enumeración Boleano asigna el calor 0 a FALSE y el valor 1 a TRUE.

5. Prototipo de función y verificación de tipo

```
1   //Funciones que no reciben argumentos
2   #include <iostream>
3
4   using namespace std;
5
6   void f1();
7   void f2(void);
8
9   int main()
10  {
11      f1();
12      f2();
13
14      return 0;
15  }
16
17  void f1()
18  {
19      cout << "La función f1 no toma argumentos\n";
20  }
21
22  void f2(void)
23  {
24      cout << "La función f2 tampoco recibe argumentos\n";
25  }
```

Funciones en línea

```
1   // Utilización de una función en línea para calcular el volumen de un cubo
2
3   #include <iostream>
4
5   using namespace std;
6
7   inline float cubo(const float s) { return s * s * s; };
8
9   int main()
10  {
11      cout << "Entre la longitud de un lado del cubo: ";
12
13      float lado;
14
15      cin>> lado;
16
17      cout << "Volumen del cubo con lado " << lado << " es " << cubo(lado) << '\n';
18
19      return 0;
20  }
```

Palabras reservadas en C y C++

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

Palabras reservadas únicamente en C++

`asm`	`catch`	`class`	`delete`
`friend`	`inline`	`new`	`operator`
`private`	`protected`	`public`	`template`
`this`	`throw`	`try`	`virtual`