

Explicación C++

• ¿Qué es una referencia?

Es una variable que recibe otro nombre para una variable ya existente que almacena la dirección de un espacio de memoria, siendo muy diferente al puntero (ya que tiene una función similar), lo que lo diferencia es que una referencia inicializada no puede hacer referencia a otro objeto o NULL, cosa que el apuntador sí. Las referencias solo se refieren a objetos ya existentes y están vinculados a estos objetos, lo que implica que no se permiten modificaciones después de la declaración. Existen dos tipos de referencias:

1. Referencias lvalue: referencia a una variable con nombre. Se expresa con "&".
2. Referencias rvalue: referencia a una variable a un objeto temporal. Se exprese con "&&".

Ejemplo

```
C++
// references.cpp
#include <stdio.h>
struct S {
    short i;
};

int main() {
    S s; // Declare the object.
    S& SRef = s; // Declare the reference.
    s.i = 3;

    printf_s("%d\n", s.i);
    printf_s("%d\n", SRef.i);

    SRef.i = 4;
    printf_s("%d\n", s.i);
    printf_s("%d\n", SRef.i);
}
```

Output

```
3
3
4
4
```

• ¿Qué es una Constante?

La constante es un valor fijo que durante toda la ejecución del programa no cambia ni puede ser cambiado, ejemplo de esto son valores estáticos que pueden ser utilizados como capacidades, constantes de operaciones entre otros. La declaración de const función miembro, esta hace que sea una función únicamente de “solo lectura” que no modifica el objeto para el que se llama. No se pueden modificar ningún miembro no estático y tampoco pueden llamar a ninguna función miembro que no comparta el tipo (Constante). Son muy útiles para especificar el tamaño de un vector facilidad de uso, confiabilidad del código y evitar el “Código Quemado”.

Para declarar una constante, se hace después de declarar las librerías y antes de las funciones, la sintaxis es la siguiente:
#define nombre_constante valor.

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

#define PI 3.1416; //Definimos una constante llamada PI

int main()
{
    cout << "Mostrando el valor de PI: " << PI << endl;

    return 0;
}
```

```
C++
// constant_member_function.cpp
class Date
{
public:
    Date( int mn, int dy, int yr );
    int getMonth() const; // A read-only function
    void setMonth( int mn ); // A write function; can't be const
private:
    int month;
};

int Date::getMonth() const
{
    return month; // Doesn't modify anything
}

void Date::setMonth( int mn )
{
    month = mn; // Modifies data member
}

int main()
{
    Date MyDate( 7, 4, 1998 );
    const Date BirthDate( 1, 18, 1951 );
    MyDate.setMonth( 4 ); // Okay
    BirthDate.getMonth(); // Okay
    BirthDate.setMonth( 4 ); // C2662 error
}
```

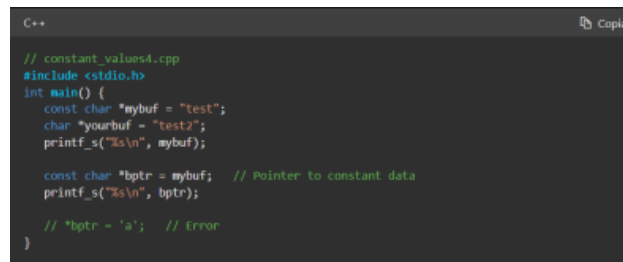
También se puede agregar constantes adentro de algunas funciones con la siguiente sintaxis: **const tipo nombre_constante.**

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

int main()
{
    const float PI = 3.1416; //Definimos una constante llamada PI
    cout << "Mostrando el valor de PI: " << PI << endl;

    return 0;
}
```

Además, también se puede hacer constante una referencia o punteros, los parámetros de una función que el cuerpo no puede modificar,



```
C++ Copiar
// constant_values4.cpp
#include <stdio.h>
int main() {
    const char *mybuf = "test";
    char *yourbuf = "test2";
    printf_s("%s\n", mybuf);

    const char *bptr = mybuf; // Pointer to constant data
    printf_s("%s\n", bptr);

    // *bptr = 'a'; // Error
}
```

pero si consultar la información del mismo y el retorno que quieres devolver un const referencia cuando devuelve una propiedad de un objeto, que no desea que se modifique fuera de él.