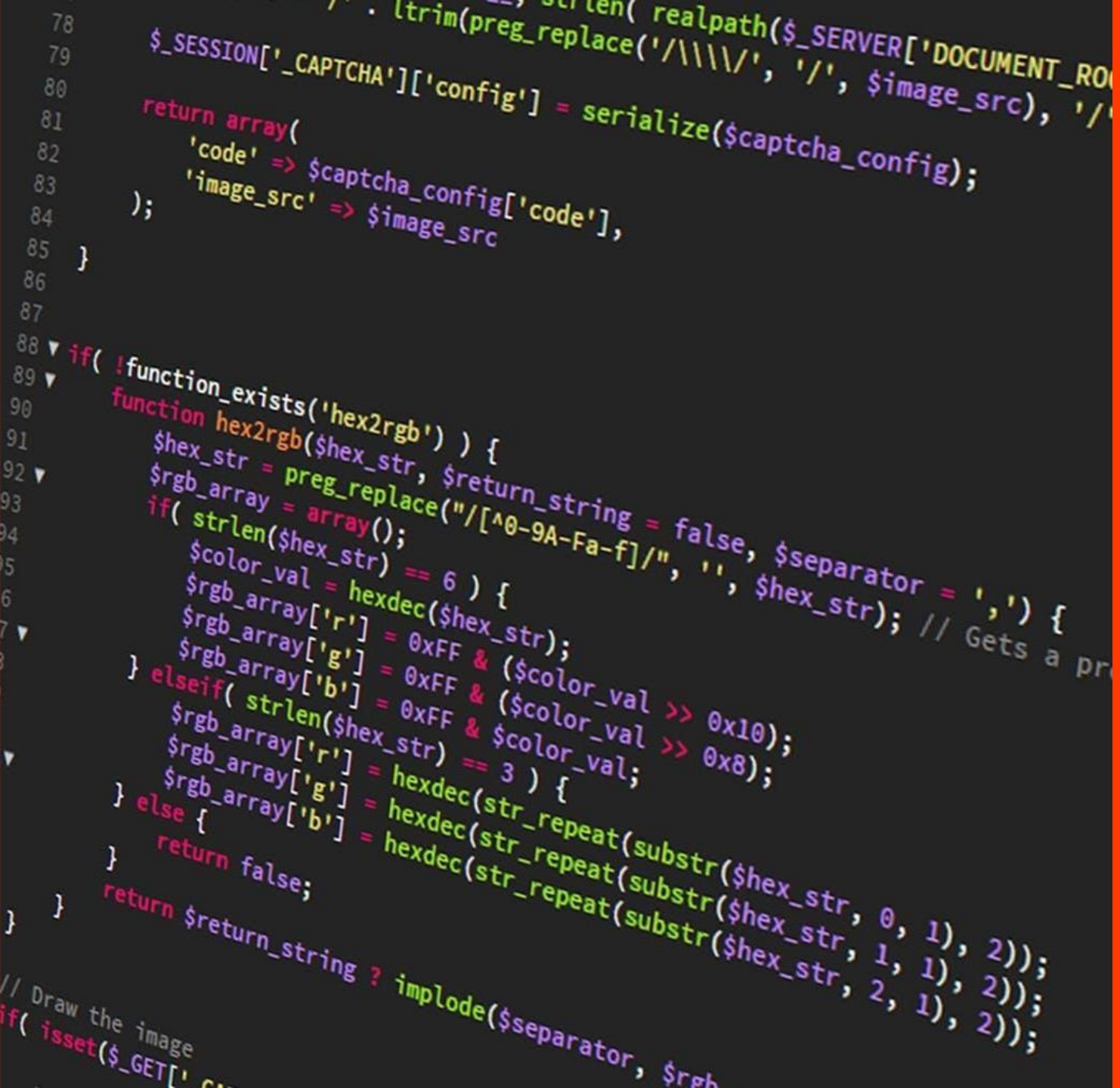
UNIVERSIDAD DE LIMA | 60 años

Paradigma Imperativo (p1)

Lenguajes de Programación


$_SESSION['_CAPTCHA']['config'] = serialize($captcha_config);
return array(
'code' => $captcha_config['code'],
'image_src' => $image_src
);
}
if(!function_exists('hex2rgb')) {
function hex2rgb($hex_str, $return_string = false, $separator = ',') {
$hex_str = preg_replace("/[^0-9A-Fa-f]/", '', $hex_str); // Gets a pr
$rgb_array = array();
if(strlen($hex_str) == 6) {
$color_val = hexdec($hex_str);
$rgb_array['r'] = 0xFF & ($color_val >> 0x10);
$rgb_array['g'] = 0xFF & ($color_val >> 0x8);
$rgb_array['b'] = 0xFF & $color_val;
} elseif(strlen($hex_str) == 3) {
$rgb_array['r'] = hexdec(str_repeat(substr($hex_str, 0, 1), 2));
$rgb_array['g'] = hexdec(str_repeat(substr($hex_str, 1, 1), 2));
$rgb_array['b'] = hexdec(str_repeat(substr($hex_str, 2, 1), 2));
} else {
return false;
}
return $return_string ? implode($separator, $rgb
}
// Draw the image
if(isset($_GET['

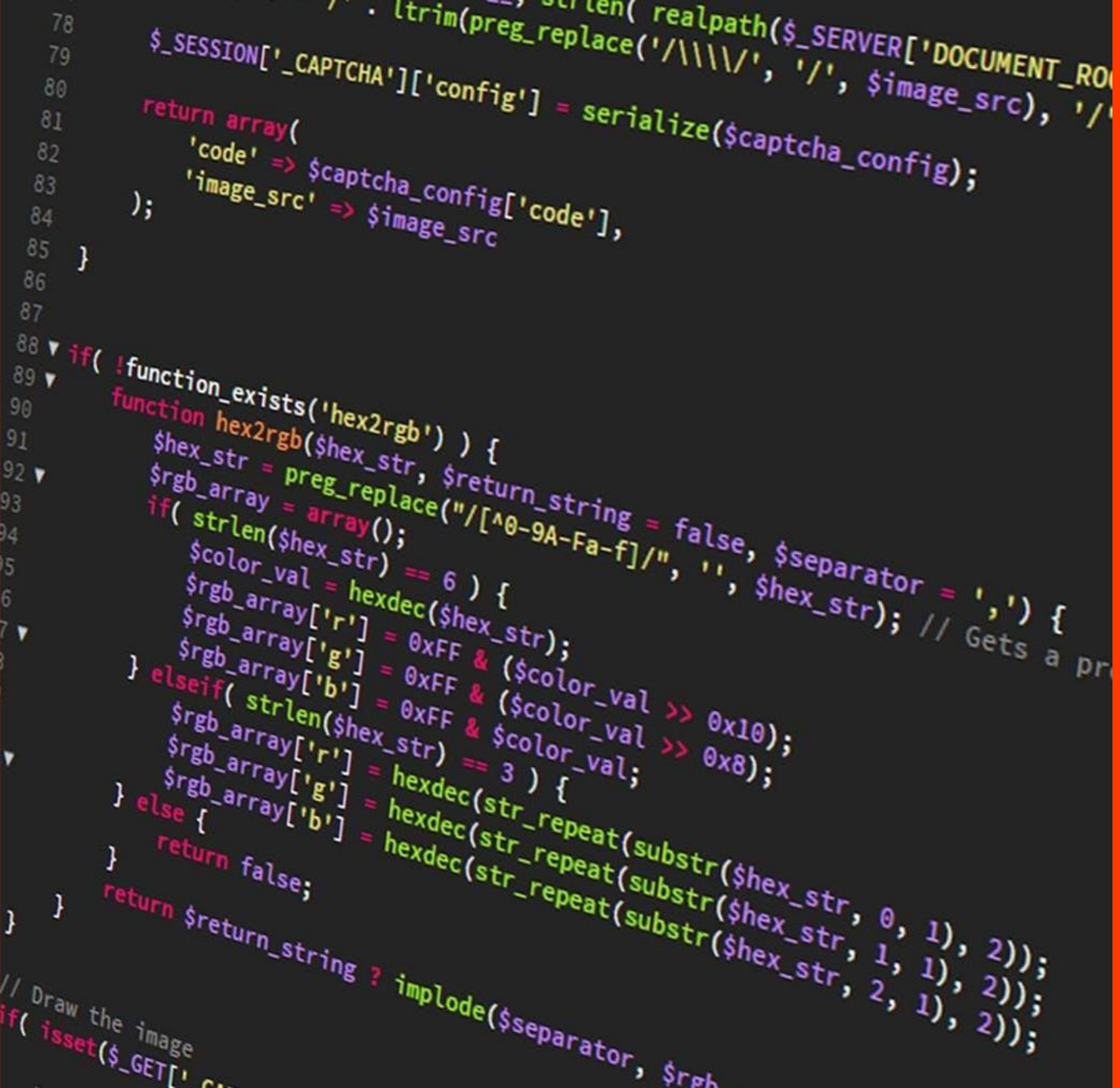
Definición

Paradigma Imperativo

- Paradigma que sigue al pie de la letra el modelo de Von Neumann.

- Siguen un flujo lineal controlado por estructuras y que trabajan con datos.

programa = estructuras de datos + estructuras de control



Tipos de Datos Simples

Tipos de datos

Un tipo de dato define el conjunto de valores que una variable o constante puede tomar.

```
int edad;
bool estaActivo;
float estatura = 1.75f;
double peso = 80.3;
const int numeroEvaluaciones = 5; // constante
```

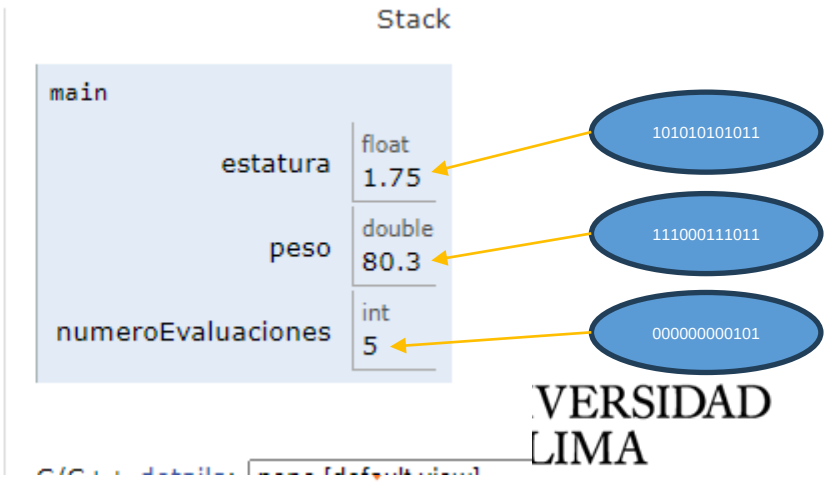
La cantidad de información que un tipo de dato puede almacenar viene definido por la plataforma donde se ejecutará el programa.

Data Type	Size
boolean	1 byte
char	1 byte
int	2 or 4 bytes
float	4 bytes
double	8 bytes

Como se ve en la visualización, la memoria “reserva” espacio para poder almacenar la información de las variables (en binario).

C++ (C++20 + GNU extensions)
known limitations

```
1  int main() {
2    int edad;
3    bool estaActivo;
4    float estatura = 1.75f;
5    double peso = 80.3;
6    const int numeroEvaluaciones = 5; // constante
7
8    return 0;
9  }
```

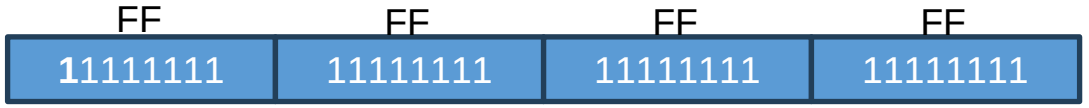


Modificadores de Tipos de datos

C/C++ nos permite definir modificadores de Tipos de Datos. Estos modificadores especifican la forma como se almacenará el valor en la variable.

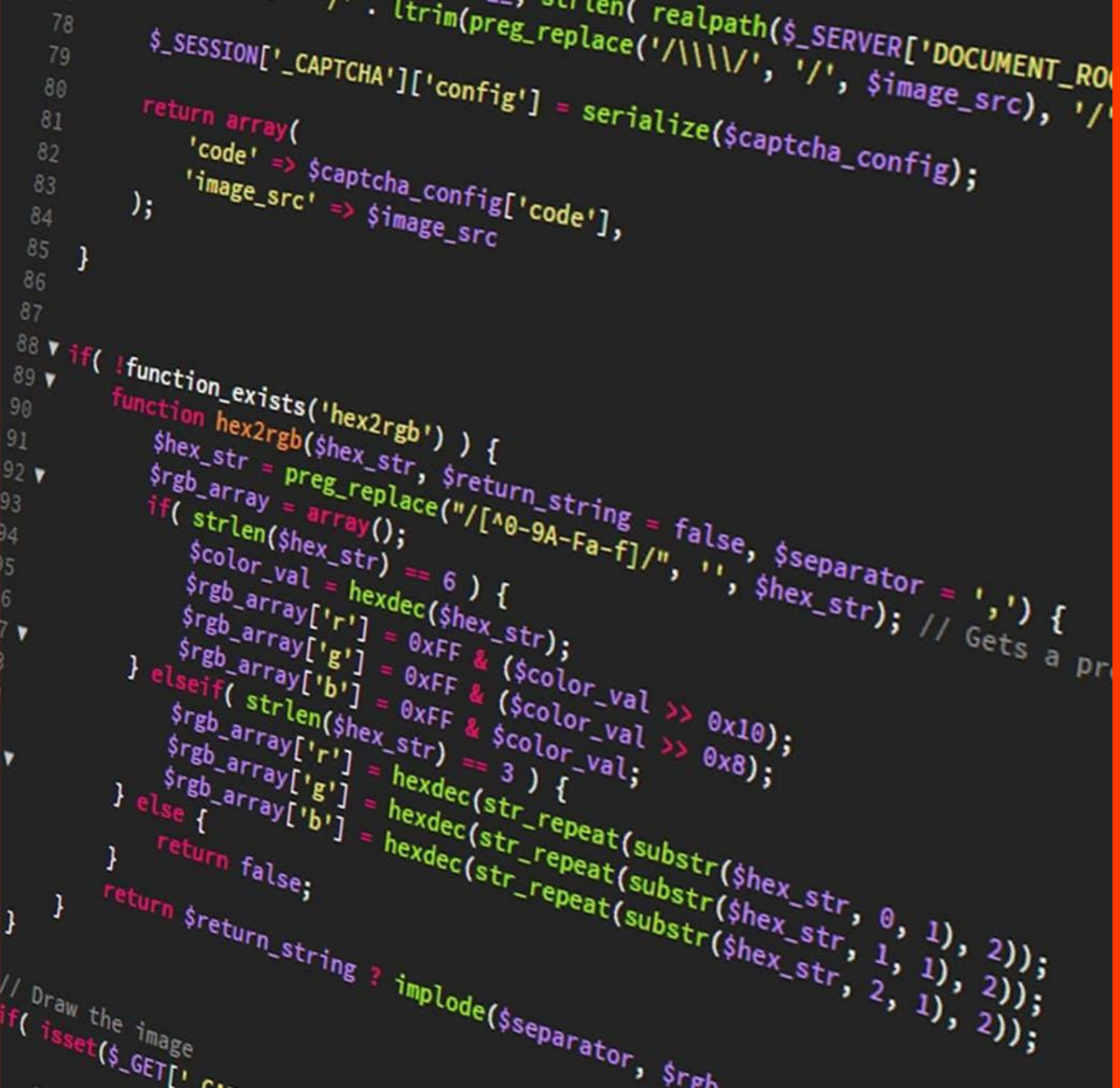
```
int maxPositivo = 2,147,483,647;
```


```
int minNegativo = -2,147,483,648;
```




```
unsigned int soloPositivo = 4,294,967,295;
```

UNIVERSIDAD DE LIMA



Conversiones (Casting)

Conversiones de tipo

Los lenguajes de programación deben darnos mecanismos de conversión entre tipos.

```
int x = 3;
x = 2.3 + x / 2
```

1. x / 2 es una división entera. Resultado = 1
2. Este resultado se suma a un double (2.3). Su resultado es un double (3.3).
3. Pero como el resultado se asigna a un int (x), el valor es "transformado" a un int.

Estos se conocen como conversiones de tipo implícitas o coercions (o *casting*).

En caso de que quiera definir directamente las conversiones, puede hacerlo utilizando paréntesis.

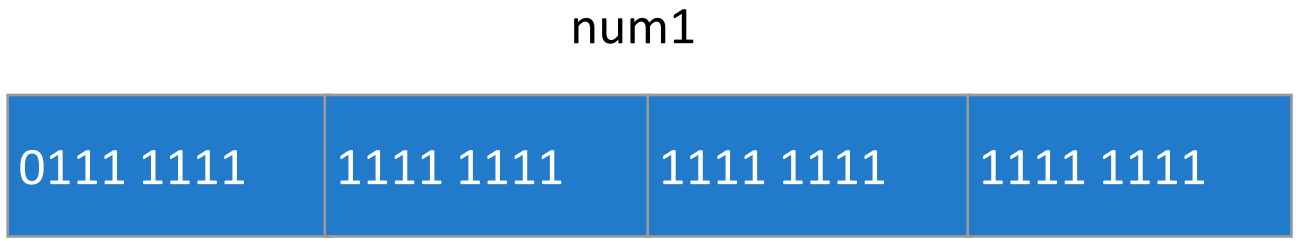
```
int x = 3;
x = ((int)2.3 + ((int)(x / 2)));
```

UNIVERSIDAD DE LIMA

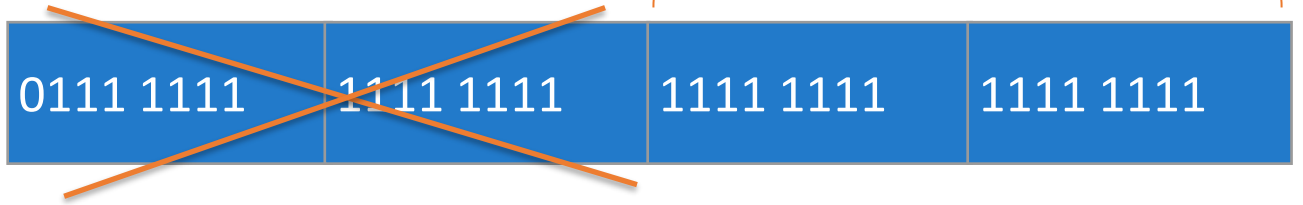
Cuidado con las conversiones

Pueden haber casos en donde el tamaño (en bytes) de los tipos de datos jueguen en contra de una buena conversión.

```
int num1 = 2147483647;
```



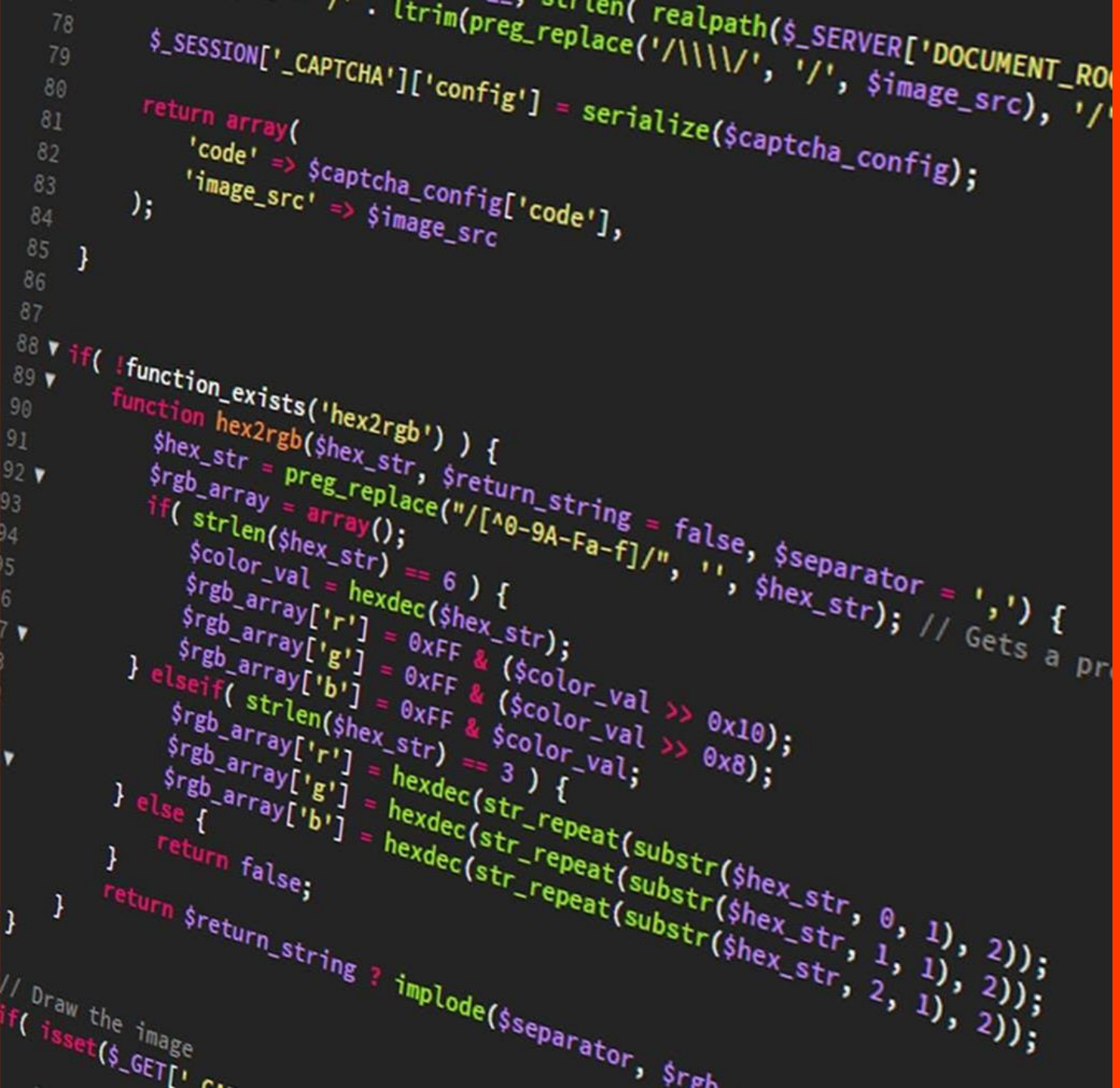

```
char num2 = (char) num1;
```



Cuál sería el valor (en decimal) de num1?

Cuál sería el valor (en decimal) de num2?

UNIVERSIDAD DE LIMA



Tipos de Datos Complejos

Enums

A veces necesitamos reducir el rango de valores que un tipo de dato puede contener.

Por ejm: utilizar un int (con rango de -2147483648 a 2147483647) que nos permite 4 294 967 296 valores para representar los días de la semana (que son 7).

Para esto utilizamos los Enums

```
enum class DiasSemana{
    Lunes = 1,
    Martes = 3,
    Miercoles = 5
};

int main(){
    DiasSemana ds;
    ds = Martes;
    printf("%d\n", ds);
}
```

UNIVERSIDAD DE LIMA

Tipos de datos complejos

Los tipos de datos primitivos (así llamado en C a los int, float, double y char) pueden formar tipos de datos más complejos.

El mecanismo de construcción de estos tipos de datos variará por lenguaje de programación. En C, los tipos de datos predefinidos que estudiaremos son los siguientes:

- Arrays
- Structs
- Funciones
- Punteros

Strings

En C no existe el tipo de dato String (cadena de caracteres). En caso quiera utilizar una cadena de caracteres, se deberá de trabajar con un arreglo de char.

```
char nombre[] = “Pepe”;
```

C++ ya implementa un tipo de dato string, pero para esto debe de importar la librería **iostream**.

```
#include<iostream>

int main()
{
   std::string nombre = "Pepe";
   std::cout << "Nombre:" << nombre << std::endl;

   std::cin >> nombre;
   std::cout << "Nombre:" << nombre << std::endl;
   return 0;
}
```

Salida del valor por la salida estándar (Shell).

Pide ingreso de un valor por la entrada estándar (Shell).

UNIVERSIDAD DE LIMA

Strings

Principales operaciones que podemos hacer con strings:

No olvidarse de incluir: `#include <string>`

Operación	Ejemplo
Obtener la longitud de un string	str.size()
Concatenar strings	string result = str1 + str2;

Convertir un string a int/double/float	int num = std::stoi(str);
	double num = std::stod(str);
	float num = std::stof(str);

Convertir int/float/double a string	std::to_string(20);
Obtener una porción de un string	std::string str = "Hello, World!"; std::string substr = str.substr(7, 5); //Extrae "World!" from the original string
Convertir a un arreglo de char	str.c_str();

Más funciones: https://cplusplus.com/reference/string/string/

UNIVERSIDAD DE LIMA

Array

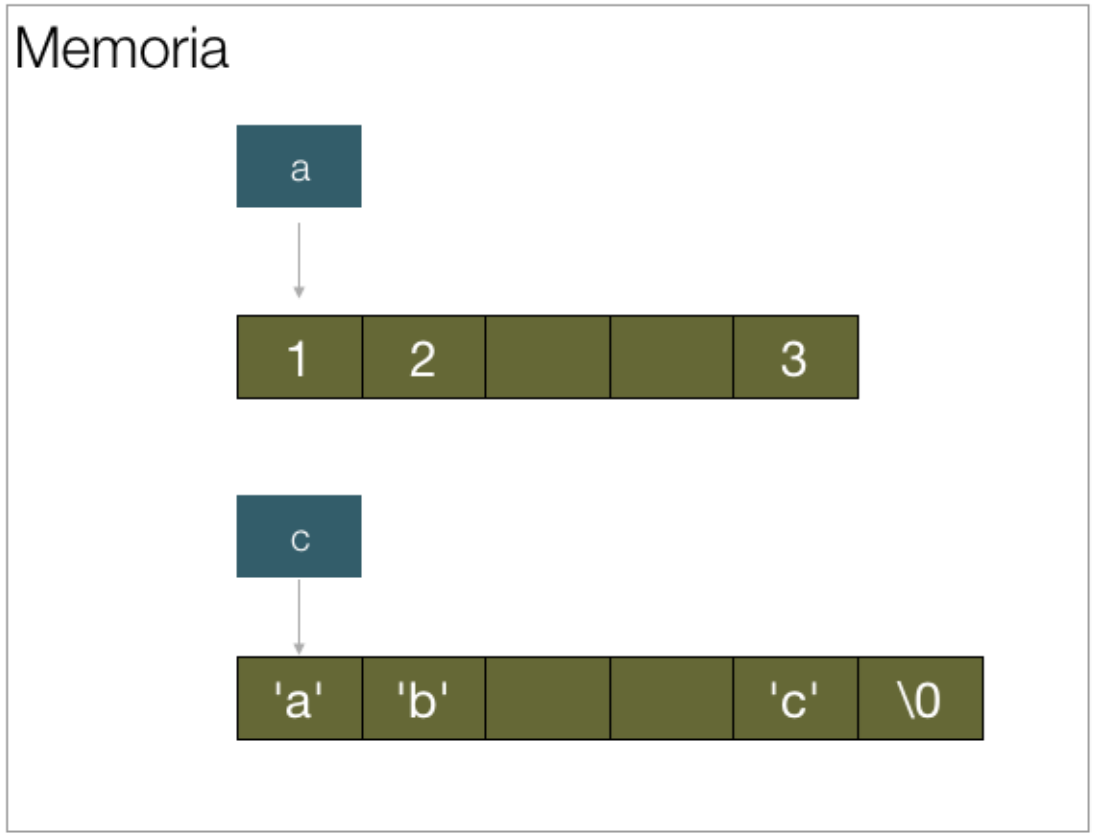
- Agrupa un conjunto de valores.

```
int a[5];

a[0] = 1;
a[1] = 2;
a[4] = 3;
a[5] = 4; ?

char c[5];

c[0] = 'a';
c[1] = 'b';
c[4] = 'c';
c[5] = 'd'; ?
```



En ciertos lenguajes hay tipos de datos parecidos como tuplas y listados.

UNIVERSIDAD DE LIMA

Structs

Nos permiten definir estructuras complejas con valores que salen del producto cartesiano de cada tipo que lo conforma.

```
struct Persona{
    char sexo;
    int edad;
    double estatura;
};


int main(){
    Persona juan;

    juan.sexo = 'M';
    juan.edad = 20;
    juan.estatura = 1.71;
}
```

Hay lenguajes donde existen tuplas que ejemplifican mejor el producto cartesiano.

UNIVERSIDAD DE LIMA

Funciones

Las funciones pueden considerarse como tipos de datos complejos.

Dependiendo del lenguaje de programación, estas pueden ser de primer nivel o no. En C, estas no son de primer nivel.

```
int maximo(int arreglo[], int tam){
    int temp, i;
    assert(tam > 0); // comprobacion
    temp = arreglo[0];
    for (i = 1; i < tam; i++){
        if (arreglo[i] > temp){
            temp = arreglo[i];
        }
    }
    return temp;
}
```

La definición de una función tiene:

- Nombre de función.
- Tipos parámetros de entrada (varios).
- Tipo de parámetro de salida (uno). Si no retorna nada retornará void.

UNIVERSIDAD DE LIMA

UNIVERSIDAD DE LIMA | 60 años

Compilación de Proyectos C++

Archivos cabecera y de implementación

- En los archivos cabecera irá la definición (NO implementación) de las funciones. Puede utilizar la extensión h o .hpp.

```
int sumar(int num1, int num2);
```

operaciones.h

- En los archivos de implementación solamente irá la implementación.
- No olvidarse de incluir los archivos de cabecera para lograr el *linkeado*. Tienen la extensión .cpp.

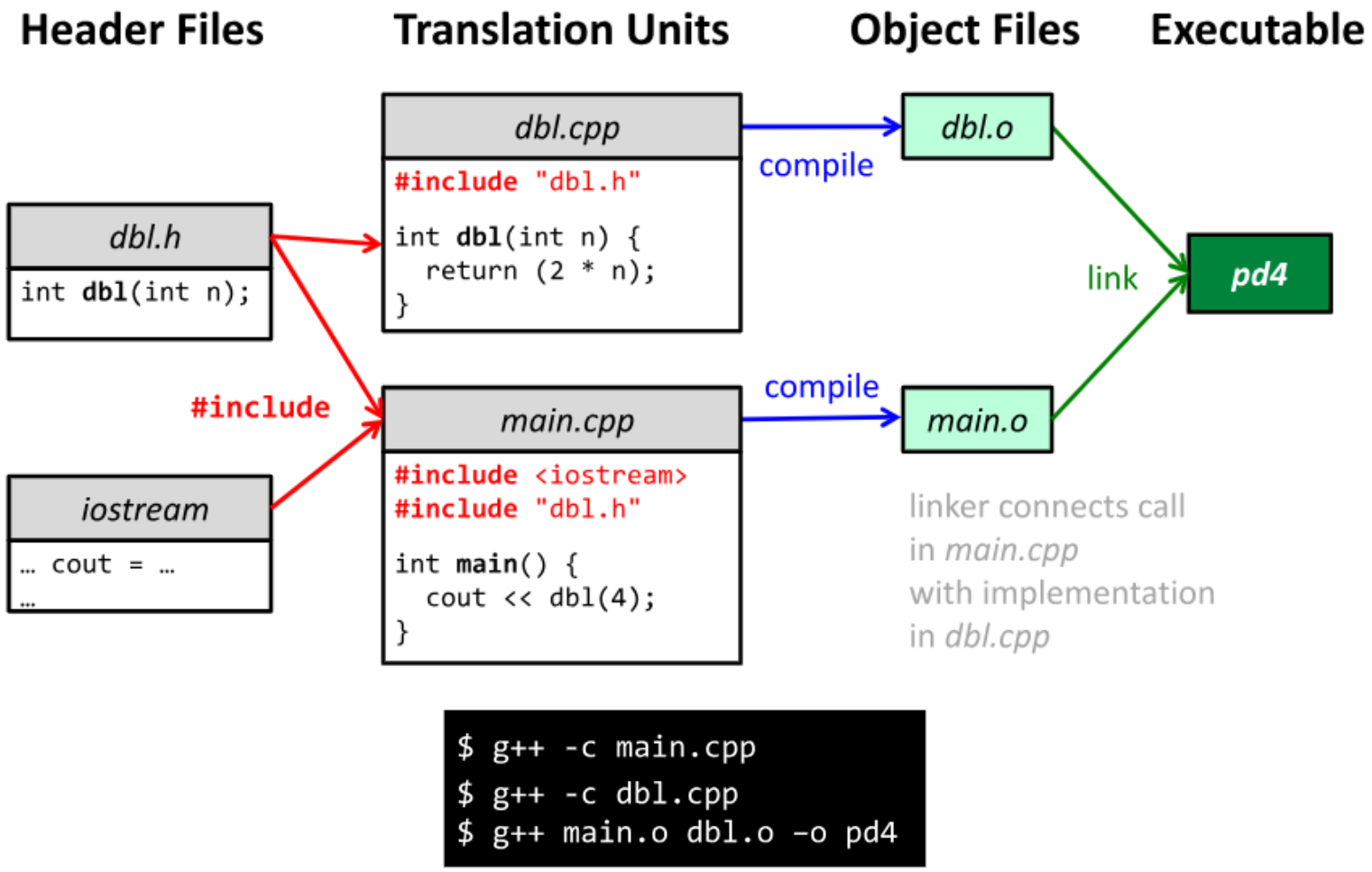
```
#include "operaciones.h"

int sumar(int num1 , int num2)
{
   return num1+num2;
}
```

operaciones.cpp

UNIVERSIDAD DE LIMA

Diagrama de Build de una Application C++




```
$ g++ -c main.cpp
$ g++ -c dbl.cpp
$ g++ main.o dbl.o -o pd4
```

Fuente: https://hackingcpp.com/cpp/lang/separate_compilation.html

UNIVERSIDAD DE LIMA

Utilizando archivo Makefile

- En el archivo Makefile vamos a poder definir las distintas tareas de construcción/ejecución que nuestro proyecto tendrá.

```
all: build

build:
   @echo "Compilando libreria"
   g++ -c operaciones.cpp
   g++ operaciones.o main.cpp -o main
   main.exe

clean:
   @echo "Eliminando"
   -rm *.exe
   -rm *.o
```

Makefile

UNIVERSIDAD DE LIMA