Introducción a C++

Juan Bekios Calfa
http://jbekios.ucn.cl
juan.bekios@ucn.cl

C y C++

C++ es la extensión de C, lanzado aproximadamente 10 años después. Aparte de soportar programación orientada a objetos, este lenguaje permite:

- Generics / templates
- Friend class / functions
- String / bool
- Un conjunto de nuevas instrucciones/librerías (ej. “stdlib.h” y “libc++.h”)
- Se puede seguir usando C en C++

NOTA 1: Ya no usa el compilador ”gcc”, ahora se usa “g++”.
NOTA 2: El estándar de C++ usado es la versión 11/14.
NOTA 3: Los archivos son “.cpp” y no “.c”.

Aspectos Generales de C++
Ejemplo 1: cout y cin

```cpp
#include<iostream>
#include<string>

int main() {
  std::string nombre;
  int edad;

  std::cout<<"Deme su nombre:"<<std::endl;
  std::cin>>nombre;

  std::cout<<"Deme su edad:"<<std::endl;
  std::cin>>edad;

  std::cout<<"Hola "<<nombre<<" cuya edad es "<<edad<<std::endl;

  return 0;
}
```

Ya no usa ".h"

Ciertas librerías son diferentes entre sistemas operativos (esto pasa también cpp)

https://repl.it/@PaulLeger/clase21

Aspectos Generales de C++
Ejemplo 2: namespace

```cpp
#include<iostream>
#include<string>

using namespace std;

int main() {
  string nombre;
  int edad;

  cout<<"Deme su nombre:"<<endl;
  cin>>nombre;

  cout<<"Deme su edad:"<<endl;
  cin>>edad;

  cout<<"Hola "<<nombre<<" cuya edad es "<<edad<<endl;

  return 0;
}
```

Namespace ahorra escribir, pero debe ser cuidadoso

No es muy buena idea usar namespace

https://repl.it/@PaulLeger/clase22

Aspectos Generales de C++
Ejemplo 3: string como objeto

```
char nombre[20] = "Jose"; //string de max 20 caracteres inicializada
char apellido[20] = "Perez";

//strcpy, copia una cadena a otra
strcpy(nombre, apellido); //Lo que hay en apellido se copia a nombre
printf("Cadena copiada: %s \n", nombre);

//strcat concatena dos cadenas
strcat(nombre, " "); //concateno con un espacio en blanco
strcat(nombre, apellido); //le añado la cadena apellido
printf("Cadena concatenada: %s \n", nombre);

system("PAUSE");
```

```
// string::length
#include <iostream>
#include <string>

int main ()
{
  std::string str ("Test string");
  std::cout << "The size of str is " <<
  return 0;
}
```

```
// string assigning
#include <iostream>
#include <string>

int main ()
{
  std::string str1, str2, str3;
  str1 = "Test string: ";    // c-string
  str2 = 'x';                // single character
  str3 = str1 + str2;        // string

  std::cout << str3  << '\n';
  return 0;
}
```

Nombres	
memcpy	copia n bytes entre dos áreas de me
memmove	copia n bytes entre dos áreas de me
memchr	busca un valor a partir de una direcc
memcmp	compara los n primeros caracteres de dos áreas de memoria
memset	sobre escribe un área de memoria con un patrón de bytes dado
strcat	añade una cadena al final de otra
strncat	añade los n primeros caracteres de una cadena al final de otra
strchr	localiza un carácter en una cadena, buscando desde el principio
strrchr	localiza un carácter en una cadena, buscando desde el final
strcmp	compara dos cadenas alfabéticamente ('a'!='A')
strncmp	compara los n primeros caracteres de dos cadenas numéricamente ('a'!='A')
strcoll	compara dos cadenas según la colación actual ('a'=='A')
strcpy	copia una cadena en otra
strncpy	copia los n primeros caracteres de una cadena en otra
strerror	devuelve la cadena con el mensaje de error correspondiente al número de error dado
strlen	devuelve la longitud de una cadena
strspn	devuelve la posición del primer carácter de una cadena que no coincide con ninguno de los caracteres de otra cadena dada
strcspn	devuelve la posición del primer carácter que coincide con alguno de los caracteres de otra cadena dada
strpbrk	encuentra la primera ocurrencia de alguno de los caracteres de una cadena dada en otra
strstr	busca una cadena dentro de otra
strtok	parte una cadena en una secuencia de tokens
strxfrm	transforma una cadena en su forma de colación (??)
strrev	invierte una cadena

https://repl.it/@PaulLeger/clase23

Aspectos Generales de C++
Ejemplo 4: bool como primitivo

```
#include<iostream>

using namespace std;

bool esPerfecto(int n) {
  bool perfecto = true;
  int suma = 0;

  for (int i = 1; i < n; ++i) {
    if (n%i == 0) suma+=i;
  }

  //return suma == perfecto; //(version corta)

  if (suma != n) {
    perfecto = false;
  }
  return perfecto;
}
```

```
int main() {
  bool perfecto;
  int suma = 0;

  int n;
  cout<<"Deme un numero:";
  cin>>n;

  //cout << boolalpha; (prueba descomentando)
  cout<<"Es perfecto "<<esPerfecto(n)<<endl;

  return 0;
}
```

¡Existe el bool!

https://repl.it/@PaulLeger/clase24

Ejemplo del uso de una estructura y new

```cpp
#include<iostream>
#include<string>

using namespace std;

struct Persona {
  string nombre;
  int numeroSocial;
  float salario;
};

typedef struct Persona TipoPersona;

int main() {

  TipoPersona* juan = new TipoPersona();

  juan->nombre = "Juan";
  juan->numeroSocial = 23;
  juan->salario = 12.3;

  cout<<"nombre:"<< juan->nombre <<endl;
  cout<<"numero Social:"<< juan->numeroSocial <<endl;
  cout<<"Salario: "<< juan->salario << endl;

  delete juan;

  return 0;
}
```

Si usa **new** con punteros, usted debe:

"juan.nombre"

por

"Juan->nombre" o "(*juan).nombre"

new solicito memoria

NOTA: **new** solo funciona en C++

delete libero memoria

https://repl.it/@PaulLeger/clase25

Scope de variables

```cpp
#include <iostream>

using namespace std;

char ch = 'D';

void scopeLexico() {
  char ch;
  ch = 'e';
  cout << " \nEl valor local de ch = " << ch << endl;

  ::ch = 'A';
}

int main() {
  scopeLexico();
  cout<< "\n\nEl valor global de ch = "<<ch<<endl;
  return 0;
}
```

"::" permitir usar el scope global

Cuidado con scope local y global de las variables

NOTA: **::** solo funciona en C++

https://repl.it/@PaulLeger/clase26

Usando new & delete

```c
#include <stdio.h>

void crearString(){
 int cant_car;
 printf(" \nCuántos caracteres en su primer nombre?");
 scanf("%d" , &cant_car);

 char *nombre = new char[cant_car +1];

 printf(" \nIngrese su primer nombre:");
 scanf("%s", nombre);
 printf("\n\nSu primer nombre, %s, es muy buen nombre.\n", nombre);

 delete[] nombre;
}

int main() {
 crearString();
 return 0;
}
```

Si reserva memoria (new), luego debe borrarla (delete)

Se puede usar funciones u otras cosas de C en C++

https://repl.it/@PaulLeger/clase27

Ejemplo new & delete

```c
typedef struct Persona Persona;

int main() {
  Persona p;
  Persona *pp;

  cout<<"direccion de p:"<<&p<<endl;
  cout<<"direccion de puntero pp:"<<pp<<endl;

  pp->edad = 32;
  cout<<"direccion de puntero pp->edad:"<<pp->edad<<endl;

  pp = &p;
  pp->edad = 32;
  cout<<"direccion de puntero pp->edad:"<<pp->edad<<endl;


  pp = new Persona();
  pp->edad = 32;
  cout<<"direccion de puntero pp:"<<pp<<endl;
  cout<<"direccion de puntero pp->edad:"<<pp->edad<<endl;

  delete pp;
  cout<<"direccion de puntero pp:"<<pp<<endl;
  cout<<"direccion de puntero pp->edad:"<<pp->edad<<endl;

  return 0;
}
```

Este código no ejecuta correctamente por motivos de punteros … ¡Investiga qué es!

https://repl.it/@PaulLeger/clase3-test

Void es “nada”

```
void f1() {
    printf("f1");
}
```

“void” significada la nada

```
void f2(void) {
    printf("f2");
}
```

Salida del Programa

```
La función f1 no tiene argumentos
La función f2 tampoco tiene argumentos
```


Tipos de datos

- **char**
- **int**
- short int (short)
- long int (long)
- **float**
- **double**
- unsigned char
- unsigned short int
- unsigned int
- unsigned long int

¿Qué es unsigned?

Operadores aritméticos

+	suma unaria, suma
-	menos unario, resta
*	multiplicación
/	división
%	resto
x=	modificar y reemplazar, donde x puede ser +, -, *, /, ó %
+ +	incremento
- -	decremento

Operadores lógicos

&&	and
||	or
!	negación
==	igual
!=	distinto
>	mayor que
>=	mayor o igual que
<	menor que
<=	menor o igual que

Operadores – Tipo Integer

Operadores Bitwise (operadores que operan bit por bit)

&	and
|	or
^	or exclusivo
~	negación
>>	desplazamiento a la derecha
<<	desplazamiento a la izquierda
x=	modificar y reemplazar, donde x puede ser &, |, ^, >>, ó <<

Operaciones a nivel de bits

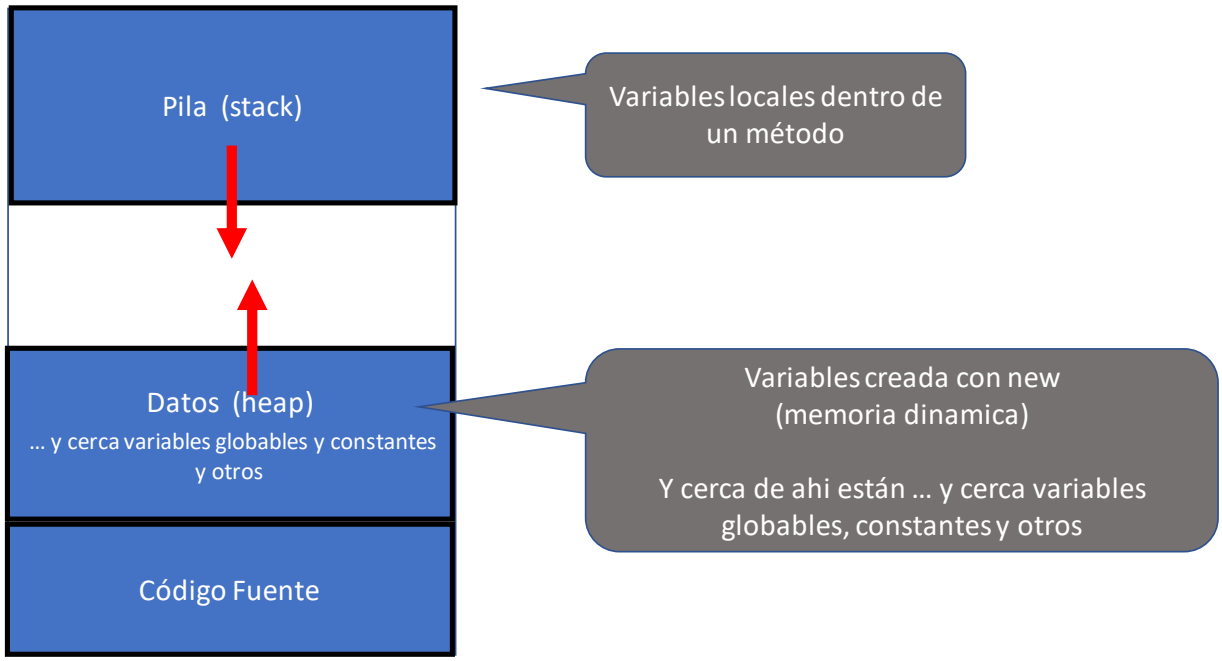
```
1   int a, b, c;
2
3   a = 0xd3; // = 11010011
4   b = 0xf5; // = 11110101
5   c = 0x1e; // = 00011110
6
7   d = a | b; // 11010011 | 11110101 = 11110111 -> 0xf7
8   d = b & c; // 11110101 & 00011110 = 00010100 -> 0x14
9   d = a ^ c; // 11010011 ^ 00011110 = 11001101 -> 0xcd
10  d = ~c;    // ~00011110 = 11100001 -> 0xe1
11  d = c << 3 // 00011110 << 3 = 11110000 ->; 0xf0
12  d = a >> 4 // 11010011 >> 4 = 00001101 ->; 0x0d
```


Aspectos Generales de C++

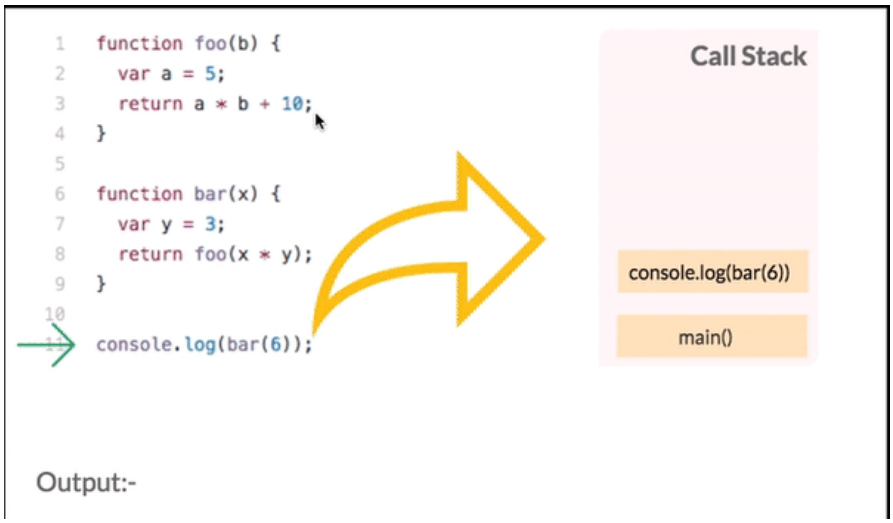
+, -	Suma y resta.
<<,>>	Desplazamiento a la izquierda, desplazamiento a la derecha.
<,>,<=,>=	Test para no igualdad
= =,!=	Test para igualdad, no igualdad.
&	AND bitwise.
^	Exclusivo OR bitwise.
|	OR bitwise.
&&	AND lógico.
||	OR lógico.
?:	**Operador condicional.**

Aprender este operador!!!!!

Programa en ejecución en la memoria (virtual)

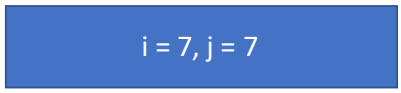


Animación del stack



Asignación Dinámica de Almacenamiento

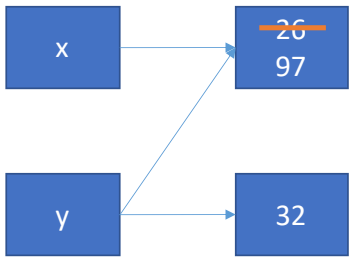
```
int i = 5;
int &j = i;
i = 7;
printf (“i = % d , j = % d”, i , j);
```



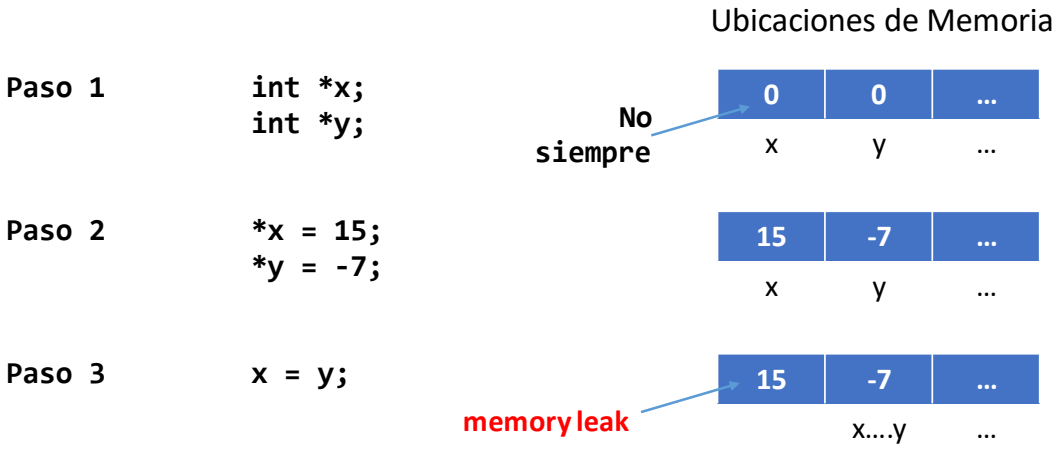
Creación de alias

```
int *x = new int [1];
int *y = new int [1];

*x = 26;
*y = 32;
y = x;
*x = 97;
```



Asignación Dinámica de Almacenamiento



Errores: Stack overflow y heap overflow

```
void infiniteLoop1(int x) {
  if (x%100 == 0) printf("%d\n",x);
  infiniteLoop1(++x);
}
```

```
void infiniteLoop2(int x) {
  if (x%100 == 0) printf("%d\n",x);
  int *c = new int[1000000000];
  //delete[] c;
  infiniteLoop2(++x);
}
```

https://repl.it/@PaulLeger/clase28

¿Cuál es “stack overflow” y “heap overflow/overrun”?

Ejemplo – Punteros y Arreglos

```
#include <stdio.h>

void imprimirArreglo(float *arr, float *fin) {
    while (arr <= fin) {
      printf ("%.0f \n",*(arr++));
    }
}


int main() {
  float *a = new float[10];
  float *fin_a = a + (10 - 1);

  printf("Comienzo memoria desde %p hasta %p \n", a, fin_a);

  for (int i = 0; i < 10; ++i) {
    a[i] = 3 * i + 12;
  }

  imprimirArreglo(a, fin_a);

  return 0;
}
```

- ¿Qué realiza “while(arr<= fin)”?
- ¿Qué falta al final del main?
- ¿Qué pasa si se cambia *(arr++) a (*arr)++?

Salida del Programa

```
12
15
18
21
24
27
30
33
36
39
```

https://repl.it/@PaulLeger/clase29