Clase #1
Paso de C a C++

Prof. Andrés Arcia
Departamento de Computación
Escuela de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Los Andes

Origenes de C

- C es un lenguaje que nació en los laboratorios Bell de AT&T en 1972 bajo las manos de Dennis Ritchie: http://www.cs.bell-labs.com/who/dmr/ y como ayuda para crear UNIX junto con Ken Thompson.
- Luego de añadir algunos tipos de datos al lenguaje B, se convertiria en los que conocemos como lenguaje C.
- UNIX fue escrito enteramente en C, entonces para usar UNIX habia que saber C.
- En 1987 aparece la version estandar según Instituto de Estandares Nacionales Americano (ANSI).
- Es bueno trabajar con el estandar, esto garantiza portabilidad entre distintas plataformas.

Variables Simples

- Para poder modelar algún problema, deben utilizarse tipos de datos:
  - *char* o tipo caracter, simbolo o letra.
  - *int* tipo entero, se alinea con la longitud de palabra.
  - *float* punto flotante de precision simple, calculos en R.
  - *doble* punto flotante de doble precision, calculos en R más extensos (precision).

Modificadores

- Pueden introducirse modificadores de tipo que agrandan o acortan la capacidad o rango:
  - unsigned, para solo números positivos
  - signed, para números positivos y negativos
  - short, reducción del rango
  - long, ampliación del rango

Ejemplos

- int k;
- long int lk;
- long otro_lk;
- unsigned char uc;

Tamaños típicos en un PC

- char 8 bits
- short int 8 bits
- int 32 bits (en un Pentium!)
- long int 32 bits
- float 32 bits
- doble 64 bits
- long double 96 bits

Pregunta

- Para qué tipos de datos tan grandes?
  - R1: El computador no es para jugar! o si?
  - R2: Aún con estos tipos hay limitaciones...
  - R3: Qué es grande?

Tipos de Almacenamiento

- Las variables locales pueden ser declaradas: **auto**, **static** o **register**.
- Variables Globales: declaradas fuera de la función; subsisten hasta que el programa finaliza. Pueden exportarse de programa en programa a través del constructo ***extern***.

Ejemplo de Variables Externas

- Haga un archivo A.cpp:

```
#include <iostream>
extern int in_other_prog;

using namespace std;

int main()
{
        cout << in_other_prog;
}
```

Ejemplo de Variables Externas

- En el ejemplo anterior, el programa no funciona porque:
  - La variable in_other_prog, no esta definida, todavia...
  - Intente compilarlo con **g++ A.cpp**, que errores obtiene?

Ejemplo de Variables Externas

- Podemos hacer un segundo programa llamado B.cpp:

```
int in_other_prog;

void change_shared_var()
{
   in_other_prog = 100;
}
```

- Observe que estamos definiendo la variable que nos hace falta... También tenemos una función que la modifica.

Ejemplo de Variables Externas

- Ahora, como juntamos este par de códigos?
  1) Compile B.cpp: g++ -c B.cpp. Así, generamos su código objeto.
  2) Compile A.cpp: g++ A.cpp -o prog B.o
- Ahora si, el compilador encontró una definición de la variable externa. Sin embargo, observe que no estamos operando sobre la variable.

Ejemplo de Variables Externas

- Intuye alguna manera de llamar a la funcion definida en B.cpp desde A.cpp?
- Qué valor imprimirá si no se hace la llamada a funcion propuesta en el punto anterior?

Variables Estaticas

- Son variables accesibles desde cualquier punto del programa sin necesidad de ser declaradas fuera de las funciones.
- Las variables estaticas no pueden ser referenciadas como externas. Es decir, solamente son validas dentro de un mismo módulo.
- Ejercicio: añada el modificador *static* al programa B.cpp y compile nuevamente. Que sucede?

Funciones

- Una función es un conjunto de instrucciones (algoritmo) que es accesible a través de una interfaz.
- Usted deberia poder predecir lo que hace la función con tan solo ver su interfaz.
- Lo anterior es cierto si Usted diseña la interfaz pensando en dicha premisa...
- Piense en ejemplos de un mal diseño...

Paso de C a C++

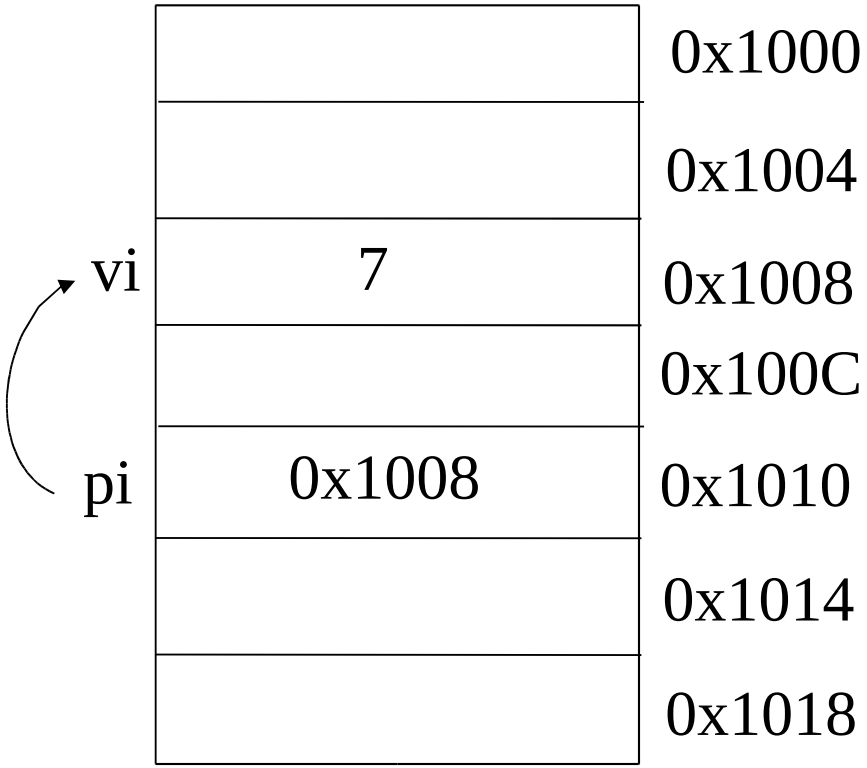
- Hacer prototipos de funciones, hablando en el sentido puritano, es la primera gran diferencia entre C y C++.
- Algunos no hacen senda distinción, y como consecuencia se obtiene una Chalupa de C/C++.
- Esto hace más difícil el paso y la comprensión de un lenguaje completamente orientado a objetos: java (por ejemplo).

Operadores

- Que son tipos de datos sin operadores?
  - R: Un Ferrari F1 sin M. Schumacher, la vinotinto sin R. Páez, etc.
- Existen varios tipos de operadores: lógicos (&&, ||), matemáticos(+, -, *), de bits(&, |), de asignación abreviada(+=) y especiales (parentesis, [], .)

Punteros en C

```
int vi;
vi = 7;
int * pi;
p=&vi
```



Aritmética de Punteros

Suponga que se tiene el siguiente arreglo de enteros:

int array[10];

3	34	6	87	95	34	56	43	35	22
0	1	2	3	4	5	6	7	8	9

La posición del primer elemento puede ser extraída como:
int * p_array = &array[0];

El n-esimo elemento se extrae como array[n-1], pero también puede utilizarse la aritmética de punteros: *(p_array + n -1).

Fíjese que p_array almacena la dirección de memoria del primer elemento y como el resto de los elementos son consecutivos, puede hacerse una simple suma o resta para llegar al elemento de interés. Note también que las unidades sumadas son relativas al tamaño.

Aritmética de Punteros

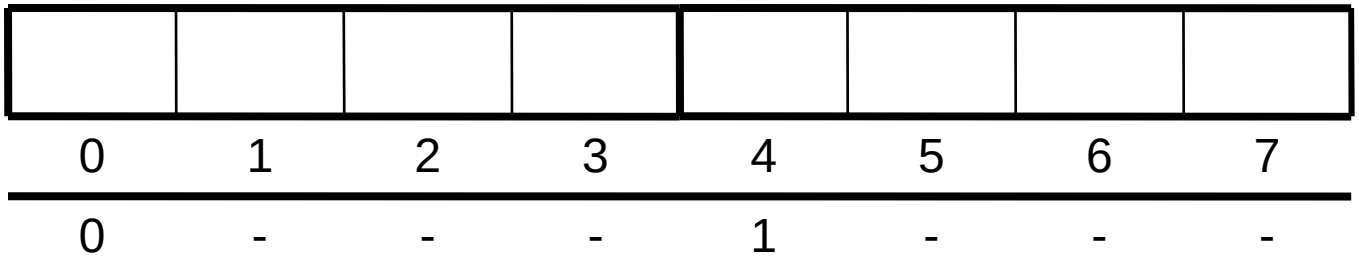
Recuerde que los apuntadores de memoria se mueven por bloques. Entonces, cuando hacemos alguna operación aritmética con algún apuntador de memoria, estamos sumando un bloque que esta formado por *m* bytes.

Ej:

```
float arr_fl[10];

int tamano_float = sizeof(float); // el tamaño de un float son 4 bytes.

float * inicio_fl;

inicio_fl = arr_fl; // esta asignación también es valida. inicio_fl apunta al principio
                    // del arreglo de flotantes.

*(inicio_fl + 5) = 11.2; // según esta asignación se esta modificando el 6to elemento
                         // del arreglo.
```

Aritmética de Punteros

Suponga ahora que se desea hacer un recorrido por cada uno de los bytes que conforma a un arreglo.



Arriba tenemos un arreglo de 2 enteros cuya declaración puede ser:

int a[2];

En la primera línea numerada tenemos la dirección de cada byte del del arreglo, que en total tiene 8 bytes (2 elem * 4 bytes_de_tamaño). En la segunda línea numerada tenemos las posiciones relativas al tipo de dato.

Aritmetica de Punteros

Si queremos navegar por todo el arreglo, *byte por byte*, podemos hacerlos de la siguiente manera:

```
#define tam_arr 10

char * p_char;

float arr_fl[tam_arr];

p_char = (char *)arr_fl;

for (int i=0; i<tam_arr*sizeof(float); i++)

        cout <<  *(p_char + i) << endl;
```

La línea que separa los arreglos y, los punteros y sus operaciones se ha difuminado!

Punteros a Funciones

- También es posible mantener funciones completas en cierta región de memoria accesible a través de su dirección.
- Declaración del tipo de la función:
  - float (* PointerType)(int, int);
  - Si se tiene: float f(int a, int b).
  - Se puede hacer: PointerType = f;
  - (*PointerType)(4,2);

Para que son buenos los punteros a funcion?

- Dan flexibilidad y extensibilidad al programa.
- Puede delegar funciones al cliente de su sistema.
- Puede implantar cosas como *callbacks.* Muy utilizado en la programación por eventos.

Ejercicios

- Haga un programa que invierta una cadena. La cadena esta dada como un arreglo de caracteres. Hagalo dentro de la misma cadena en una primera versión, en otra versión en una cadena nueva y en otra versión en una cadena que se asume viene lista para ser escrita pero distinta a la de entrada.
- Haga un programa que ordene un arreglo de enteros.

Consideraciones:

Ambos programas contener una función que realice la tarea exigida. Piense cuidadosamente la interfaz: nombre de la función y parámetros.