Estructura de Datos y Algoritmos
Departamento de electrónica
Universidad Técnico Federico Santa María

DEPARTAMENTO DE ELECTRONICA

Ayudantía 1: Conexión a Aragorn e introducción a C

Objetivo Principal: Introducir a conceptos básicos de C.
Autor: Kevin Pizarro Aguirre

---

1. Introducción a C

Comparación Python vs C

Ya conociendo un lenguaje de programación de alto nivel, como lo es Python, ir a un lenguaje de programación de nivel medio es análogo a excavar más profundamente, conocer y manipular los elementos de interés más en detalle. Por ejemplo, al comparar el típico programa de "Hello world" es posible notar que en Python se puede realizar en 1 línea mientras en C se debe realizar en mínimo 3 líneas.

```python
print("Hello there...")
```

Listing 1: "Hello world" en Python.

```c
#include <stdio.h>
int main() {
    printf("Hello there...");
    return 0;
}
```

Listing 2: "Hello world" en C.

En C podemos manipular la función main para que retorne un 0 y saber que el programa terminó sin complicaciones, además para poder imprimir por pantalla se utiliza la biblioteca "stdio.h" la que permite utilizar la función printf. De esta forma se manipula qué funciones queremos utilizar para no "desperdiciar" espacio en memoria.

Por otro lado, se puede notar que en C debemos explicitar cuál es la función principal o función main mientras que en Python intuye que el programa en sí es el principal y por ende el cuál ejecutará. Quedará aún más en evidencia que Python suele "asumir" mucho sobre nuestro código, mientras que en C quedará todo en nuestras manos y no debería quedar duda sobre qué está realizando efectivamente el programa desde una mirada más profunda.

Conceptos y operaciones básicas en C

Ya se ha visto el programa de prueba por defecto ("Hello world"), pero se tomará otros ejemplos con más características para ver más en detalle cada línea, su uso, sus funciones y los conceptos detras de ello.

Link al repositorio con los códigos para cada tema y otros recursos útiles. Se podrán ver códigos con los temas: Directivas de pre-procesamiento, algunas funciones de la librería stdio.h, comentarios en línea y en bloque, tipos de datos y calificadores, finalmente control de flujo y operadores básicos.

Directivas de preprocesamiento

Son aquellas sentencias del tipo **#include <stdio.h>** y **#define SIZE 5**. Son ejecutadas antes de la compilación, la primera sentencia nos permite utilizar las funciones de la librería stdio (standard input/output) y la segunda realizar una definición de un "parámetro" SIZE con valor 5 (esta forma no ocupa espacio en memoria para SIZE). En Python sería análogo a realizar un **import numpy as np**, por ejemplo.

Funciones de la librería stdio.h

Se tomarán sólo algunas de ellas. Para mayor detalle buscar en la documentación y en el apéndice del libro guía en la sección correspondiente a las librerías estándar.

- printf(const char *format, ...): Imprime una salida con formato a la consola (stdout).
- scanf(const char *format, ...): Lee una entrada con formato deesde la consola (stdin).
- gets(char *str): Lee una línea desde la consola y la almacena en el puntero str. Termina cuando se alcanza un carácter de salto de línea o cuando se alcanza el termino de archivo (EOF).
- fopen(const char * filename, const char * mode): Crea o abre el archivo indicado, en los modos indicados.
- fclose(FILE *fp): Cierra el archivo indicado.
- fputs(const char *s, FILE *fp): Escribe el string s en el archivo indicado, en la posición que apunte fp.
- fgets(char *buf, int n, FILE *fp): Lee n-1 caracteres desde la posición de fp. Copia el string leído en buf, agregando el carácter nulo al final.

Comentarios en línea y en bloque

Los comentarios en línea implementan con el operador **//**, mientras que los comentarios en bloque con **/*bloque*/**. En general los comentarios en línea sirven para comentarios cortos sobre una línea en específico, mientras que en bloque para comentarios más extensos por ejemplo sobre el mismo archivo o funciones. Recordar que un comentario sólo es una ayuda para el programador, el compilador o la máquina en sí los omite. Para Python, los comentarios se realizan con el operador #.

Tipos de datos y calificadores

Los tipos de datos primitivos en c son: **int**, **float**, **double** y **char**. Luego utilizando calificadores se le puede dar características más específicas por ejemplo que ocupe más espacio (implica mayor rango de datos) o que sean con o sin signo. Los calificadores son: **short**, **long** y **unsigned**, con ello se pueden realizar las combinaciones que se estimen convenientes. Ejemplos de combinaciones

- short int: Número entero que utiliza 2 bytes de memoria. (int ocupa 4 bytes)
- unsigned long int: Número entero sin signo (número positivo) que utiliza 8 bytes de memoria.
- unsigned char: Carácter sin signo (¡¿QUÉ?!). Recordar que un caractér puede ser representado numéricamente por la tabla ASCII.

Notamos que Python se abstrae de aquello, queda a su interpretación que tipo de dato queremos utilizar.

Control de Flujo y operadores básicos

Dentro de los operadores podemos encontrar los condicionales y los aritméticos.

Operadores condicionales: Son aquellos que comparan dos valores y nos retornan el valor de verdad, un 1 en caso de ser verdadero (true en Python) y un 0 en caso de ser falso (false en Python). A continuación los operadores.

- < : Menor que.
- > : Mayor que.
- <= : Menor o igual que.
- >= : Mayor o igual que.
- == : Igual que.
- ! = : Distinto que.
- && : Operador **Y**, equivalente a **and** en Python.
- || : Operador **O**, equivalente a **or** en Python.
- !: Niega el valor de verdad. Si es falso pasa a ser verdadero.

Operadores aritméticos: Son aquellos que modifican el valor de cierta variable o nos entregan un resultado. A continuación los operadores.

- + : Suma los valores de dos variables.
- - : Resta los valores de dos variables.
- * : Multiplica los valores de dos variables.
- / : Divide los valors de dos variables.
- % : Entrega el módulo o el resto de la división entre dos variables.

Además se tiene el operador de asignación **=**, para darle valor a alguna variable.

Control de flujo y bucles Existen varias sentencias de control de flujo y bucles que son compartidas con otros lenguajes de programación. A continuación una lista de los más útiles.

- if(condición) : Si la condición es verdadera entonces se ejecuta el bloque de código dentro del if.
- else : Se utiliza luego del if, de esta forma le damos la opción **sino**. Es decir si no se cumple la condición del if, entonces se ejecutará el bloque de código del else.
- switch-case: Verifica a cual de los casos corresponde la condición, en caso que ninguno cumpla se ejecuta el caso por defecto (si lo hubiese).
- bucle for(valores iniciales; condición; operación al final del bucle): Antes que todo inicia los valores indicados, luego al inicio de las iteraciones verifica la condición, si la cumple entonces ejecuta el bloque de código dentro del for, en caso contrario lo termina. Finalmente ejecuta las operaciones del último parámetro.
- bucle while(condición): Mientras se cumpla la condición se ejecutará el bloque de código.
- break: Esta sentencia nos permite romper el bucle en el cual se use.
- continue: Esta sentencia "salta" la iteración actual y pasa a la siguiente.

2. Ejercicios

1. Escriba un programa en C que imprima por consola un mensaje de bienvenida utilizando la función printf. Finalmente compile el programa que acaba de escribir ejecútelo para confirmar su funcionamiento.

2. Escriba un programa que le permita repetir un mensaje de entrada por consola pero en upper case. Por ejemplo, el usuario ingresa “Hola caracola” y como salida debe entregar “HOLA CARACOLA”.

3. Escriba un programa que calcule un promedio simple para su ramo favorito (EDA). Las notas de los certámenes serán recibidas por la consola. El programa termina cuando se ingresa el carácter ’q’, asumiendo que sólo se ingresarán números enteros entre 0 y 100. ¿Cómo modificaría el programa para agregarle ponderaciones al promedio de tareas y al promedio de certámenes?

4. Escriba un programa que imprima por consola un triángulo rectángulo hecho con *. La entrada por consola debe ser la cantidad de filas o la “altura” del triángulo. ¿Qué modificaría si se desea imprimir una pirámide o un diamante en vez del triángulo?

5. Escriba un programa que invierta una cadena de caracteres. Ejemplo, si se recibe “Severla” por consola entonces se debe imprimir “alreveS”. Apóyese de la librería string.h si encuentra pertinente.

6. Escriba una calculadora de interés compuesto. Recuerde que la fórmula es $VF = VI(1 + i)^n$, donde VF es el valor final, VI el valor inicial, i la tasa de interés y n el número de periodos. Asuma que la tasa de interés y el número de periodos ingresados son coherentes en cuanto unidad de medida.

Nota: Las soluciones de los códigos se encuentran en el repositorio en la sección de Ayudantias -> Codigos -> Resolucion_Ejercicios.