Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3641 – Lenguajes de Programación I
Abril–Julio 2022

Examen 1 (25 puntos)

Christopher Gómez, #18-10892

1. Escoja algún lenguaje de programación de alto nivel y de propósito general cuyo nombre empiece con la misma letra que su nombre.

(a) De una breve descripción del lenguaje escogido.

Lenguaje C:

Es un lenguaje de programación imperativo y procedural de propósito general creado en 1972 por Dennis Ritchie, inicialmente para diseñar utilidades para el sistema operativo Unix, siendo la implementación de sistemas operativos uno de sus principales propósitos. Es de alto nivel, tipado estático y débilmente tipado.

Es un lenguaje compilado, que provee acceso de bajo nivel a la memoria. Fue estandarizado por primera vez en 1983, conocido como ANSI C, tras su primera iteración se han publicado nuevos estándares, como C99 el año 1999, y C11 y C17, en 2011 y 2017 respectivamente. Se estará haciendo referencia en este caso al estándar adoptado en 1999 (C99).

i. Diga qué tipo de alcances y asociaciones posee, argumentando las ventajas y desventajas de la decisión tomada por los diseñadores del lenguaje, en el contexto de sus usuarios objetivos.

El lenguaje C ofrece alcance léxico o estático, lo que es decir que la resolución de nombres depende de la ubicación de las variables en el código fuente y no del estado del programa a tiempo de ejecución. La ventaja de esto es que es más fácil de razonar sobre el código, ya que no fuerza al programador a pensar sobre la ejecución del programa para determinar cuál será el entorno de referencia en un momento dado, sino que puede saberse viendo la estructura del código. Por otro lado, al ser un lenguaje inicialmente orientado a la implementación de sistemas operativos, su filosofía se basa en buscar la eficiencia manteniendo la sencillez, y como el alcance estático permite resolver referencias a tiempo de compilación, esto se traduce en programas más eficientes, que dejan la menor cantidad de trabajo posible al tiempo de ejecución, crítico en el contexto de sus usuarios objetivos (usado como estándar hoy en día, además de para la implementación de sistemas operativos, también para sistemas embebidos, microcontroladores, manejadores de dispositivos, etc., en los que es muy importante el buen desempeño).

Por otro lado, en C las funciones no son de primera clase, ya que aunque existe forma de asignar funciones a variables, pasarlas como argumentos y devolverlas en funciones, esto se logra con el uso de apuntadores, que son de primera clase, y en particular pueden hacer referencia a direcciones de memoria que ocupan funciones, que pueden ser derreferenciadas luego y usadas. Sin este método, no es posible tratar a las funciones como ciudadanos de primera clase; además, tampoco permite la creación de funciones anónimas ni la anidación de funciones. Por estas razones, C no cuenta con ninguno de los dos tipos de asociación, ya que al no permitir que las funciones sean de primera clase no existe el problema en determinar entornos de referencia, por lo que no implementa la creación de clausuras.

ii. Diga qué tipo de módulos ofrece (de tenerlos) y las diferentes formas de importar y exportar nombres.

En C no existen los módulos como una abstracción, es decir, el lenguaje no ofrece mecanismos nativos para importar o exportar nombres sin contaminar el espacio de nombres, sin embargo, estos pueden ser emulados mediante directivas de preprocesador y técnicas de compilación separada. Se incluye el código fuente de otro archivo usando #include “<ruta>”, y generalmente se separa el código en una cabecera (archivos .h, de header), que es un archivo con los prototipos (declaraciones) de variables y funciones que podrá ser accesadas en otro código, y cuerpo (archivos .c), donde se incluye la implementación de las funciones, logrando así una especie de abstracción parecida a la de los módulos.

iii. Diga si el lenguaje ofrece la posibilidad de crear aliases, sobrecarga y polimorfismo. En caso afirmativo, dé algunos ejemplos.

C ofrece la posibilidad de crear aliases, es decir, de tener más de un nombre para referirse al mismo objeto del lenguaje. Uno de los ejemplos más claros es el de la palabra clave del lenguaje `typedef`, que permite crear un alias a los tipos del lenguaje o los definidos por el usuario, su gramática es: `typealias <tipo de datos> <nuevo nombre>;` así, hacer `typealias unsigned long int u32;` permite referirse al tipo `unsigned long int` como `u32`, sin descartar el nombre anterior, pudiendo referirnos al mismo tipo mediante dos nombres distintos.

Otro de los ejemplo más comunes y claros de aliasing en C son los apuntadores de memoria, ya que se pueden tener varios apuntando a la misma dirección de memoria, de tal forma, considérese el siguiente programa:

```
#include <stdio.h>
int main () {
      int x = 1;
      int *y = &x;
      *y = 8;
      printf(“%d”, x);
}
```

Se declara x y luego y como un apuntador al espacio de memoria donde reside, luego con el operador * se derreferencia y, asignando 8 a la memoria apuntada por esta. Este programa imprime `8` al imprimir x, aunque el cambio se hizo mediante la variable y, por lo que podemos decir que efectivamente hemos creado dos nombres distintos para hacer referencia al mismo objeto en particular.

Por otro lado, C no soporta sobrecarga de ninguna forma como interfaz externa, es decir, el usuario no puede definir funciones sobrecargadas para parámetros de diferentes tipos y distinta cantidad de los mismos, sin embargo, existen funciones variádicas, como printf, que reciben una cantidad de parámetros variables, sin embargo, esto no representa una forma de sobrecarga. En cierto sentido, ciertos operadores están sobrecargados, como por ejemplo el de suma '+', que se puede usar para sumar int, long, float etc. (int x = `1 + 2; float y = 0.1 + 0.1;`), sin embargo, es esta la única forma de sobrecarga que existe en el lenguaje, no estando disponible como posibilidad en la capa de abstracción funcional.

Finalmente, C tampoco ofrece soporte intrínseco para polimorfismo de ningún tipo, pudiendo este comportamiento ser simulado mediante patrones de diseño, como los son estructuras con arreglos y tablas de apuntadores a funciones, que aunque puede emular polimorfismo, no se tratan de una interfaz uniforme. En C99 existe la librería `tgmath.h`, que define funciones matemáticas de tipo genérico, que implementa una especie de polimorfismo con el uso de macros, sin embargo, no es posible para el usuario definir sus macros para emular el comportamiento deseado sin usar extensiones de compilador, en vez de características nativas del lenguaje.

iv. Diga qué herramientas ofrece a potenciales desarrolladores, como: compiladores, intérpretes, debuggers, profilers, frameworks, etc.

Al ser uno de los lenguajes más populares y antiguos, no es poca la variedad de herramientas disponibles para C hoy en día, estas son algunas de ellas:

- Compiladores: Existe una gran variedad de estos, los más populares y usados son GCC (GNU Compiler Collection) y Clang (los más extendidos), Tiny C Compiler, Portable C Compiler, Microsoft Visual C++, entre otros.
- Intérpretes: Aunque C es un lenguaje compilado, existen implementaciones de intérpretes para C, algunas de ellas son picoC y Ch, que ofrecen la posibilidad de hacer scripting con C.
- Depuradores: Los más utilizados son GBD (o GNU Debugger) y Valgrind, el primero ofrece la posibilidad de detener la ejecución de detener el programa en un punto, ver el estado de las variables, modificarlas y avanzar paso a paso, entre otras posibilidades, mientras que el segundo es más usado para depurar problemas relacionados con el uso de memoria (fugas de memoria, punteros colgantes…) y la eficiencia de los programas.
- Profilers: Microsoft ofrece el Visual Studio Team System Profiler, Valgrind es muchas veces referido como un profiler, existen otros como gprof, Google Performance Tools, KCacheGrind, etc.

- Frameworks y librerías: Existen numerosas librerías para diversos propósitos, OpenGL y SDL proporcionan soporte para gráficos, Tk para la creación de interfaces gráficas multiplataforma, plplot y gnuplot para graficado, etc.

(b) Las implementaciones de esta y las demás preguntas se encuentra organizada por directorios en el siguiente repositorio:

https://gitfront.io/r/user-1303949/rogvsVoc6rSG/Parcial1-CI3641-AJ2022-Chus/

Cada directorio contiene un README.md con instrucciones sobre su ejecución y cualquier información adicional. El código fue probado en el sistema operativo Debian 11 Bullseye.

2. Adjunto al final

Nota: En la resolución de este ejercicio se usan las palabras “función” y “subrutina” de forma intercambiable (por costumbre y que no me di cuenta al exportar los PDF :/), ambas se refieren a las subrutinas del programa dado.