EXAMEN PRÁCTICAS (MIPS)
ARQUITECTURA DE COMPUTADORES
3º, IST, ITT, IT, 4º IT-ADE, IT-AEROESPACIAL URJC
ARQUITECTURA DE SISTEMAS AUDIOVISUALES II
4º SAM, URJC

Fuenlabrada, 10 de Junio de 2019

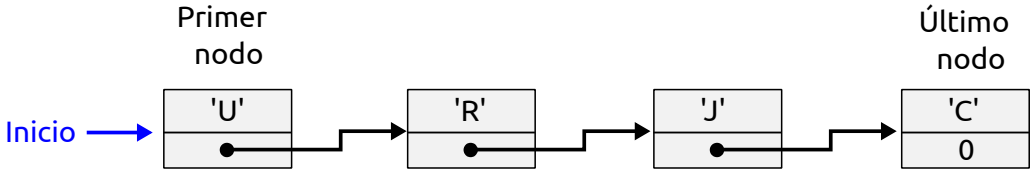
AVISO: Asegúrate que tus programas **cumplen** con los siguientes criterios. Si no se cumple alguno de ellos la **nota máxima** de tu examen será de **2 puntos**

- **Cumplimiento de especificaciones**. Se deben cumplir las especificaciones indicadas en el enunciado: nombres de funciones, nombres de archivos, funcionalidad, número de parámetros, etc. Compruébalo antes de entregar el examen
- **Respetar el convenio**. Resuelve las preguntas **sin violar** el convenio del uso de registros
- **Sin errores en tiempo de ejecución** (Runtime errors). Tus programas no deben generar excepciones al ejecutarse

Queremos almacenar caracteres usando una **lista simplemente enlazada**, con la que representaremos **cadenas de caracteres**. La estructura para representar los nodos de la lista consta de una palabra para almacenar el carácter y un puntero al siguiente nodo. La representación en lenguaje C es la siguiente:

```
typedef struct _node_t {
   int car;               /* Caracter. Tamaño: Palabra */
   struct _node_t *next;  /* puntero al nodo siguiente */
} node_t;
```

De esta forma, la cadena “URJC” querería representada en memoria por esta estructura:



El nodo que contiene el último carácter siempre apunta a **null** (la dirección del campo next es 0). **Inicio** es un puntero al primer nodo de la lista.

Se pide implementar las **siguientes funciones** y el **programa principal** indicado:

1. **Función create** (1 Ptos): **`node_t * create(int car)`**: Crea un nuevo nodo. Inicializa el campo car con el valor pasado como argumento. El campo next se inicializa a 0. Devuelve la dirección del nodo creado

2. **Función print_node** (1 Ptos): **`void print_node(node_t *nodo)`**: Imprime en la consola el carácter contenido en el nodo apuntado por el parámetro nodo. No devuelve nada

3. **Programa principal. Parte I** (1 Pto): El programa principal debe comenzar comprobando que las funciones create y print_node funcionan correctamente. Se debe hacer lo siguiente:

- Crear un nodo que contenga el carácter '>', llamando a **create**
- Imprimir ese carácter llamando a la función **print_node**
- Este será nuestro nodo de inicio: es el primer carácter de nuestra cadena. Se debe almacenar su dirección en el **registro s0**

4. **Función insert_node** (3 Ptos): `void insert_node(node_t *inicio, int car)`: Crea un nodo nuevo inicializado con el carácter *car* y lo inserta al **final** de la lista. El primer argumento es el puntero al comienzo de la lista. Debe llamar a la función **create**. El algoritmo debe ser **iterativo**

5. **Función print** (2 Ptos): `void print(node_t *inicio)`: Imprimir la cadena completa, almacenada en la lista. Se pasa como argumento el puntero al comienzo de la lista. La función recorre la lista de forma **iterativa** imprimiendo cada carácter llamando a la función **print_node**

6. **Programa principal. Parte II** (2 Ptos): A continuación de la parte I, el programa principal debe **pedir al usuario** caracteres individuales, e irlos insertando en la lista que había sido creada (cuya dirección del primer elemento estaba almacenada en s0), hasta que se introduce el carácter '\n' (tecla enter). Este carácter NO se mete en la lista. Luego se imprimirá la cadena completa, llamando a la **función print()**. El programa termina.