Primer Parcial

Primer Cuatrimestre 2022

Normas generales

- El parcial es INDIVIDUAL
- Una vez terminada la evaluación se deberá completar un formulario con el *hash* del *commit* del repositorio de entrega. El link al mismo es: `https://forms.gle/2o2Xnt6iVsPrZX1s9`.
- Luego de la entrega habrá una instancia coloquial de defensa del trabajo

Compilación y Testeo

El archivo `main.c` es para que ustedes realicen pruebas básicas de sus funciones. Sientanse a gusto de manejarlo como crean conveniente. Para compilar el código y poder correr las pruebas cortas implementadas en main deberá ejecutar `make main` y luego `./runMain.sh`.
En cambio, para compilar el código y correr las pruebas intensivas deberá ejecutar `./runTester.sh`. El programa puede correrse con `./runMain.sh` para verificar que no se pierde memoria ni se realizan accesos incorrectos a la misma.

Pruebas intensivas (Testing)

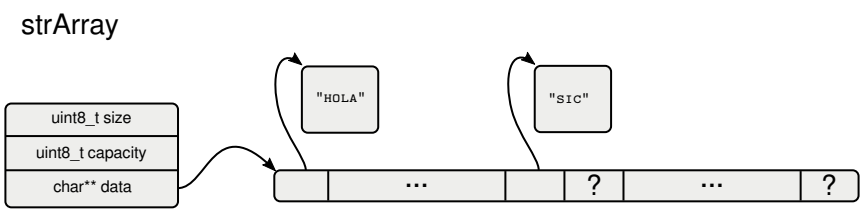
Entregamos también una serie de *tests* o pruebas intensivas para que pueda verificarse el buen funcionamiento del código de manera automática. Para correr el testing se debe ejecutar `./runTester.sh`, que compilará el *tester* y correrá todos los tests de la cátedra. Luego de cada test, el *script* comparará los archivos generados por su parcial con las soluciones correctas provistas por la cátedra. También será probada la correcta administración de la memoria dinámica.

Enunciado

Estructura `strArray`

Implementa un array dinámico de capacidad limitada. El mismo puede contener como máximo la cantidad de strings indicada en `capacity`. Los datos serán todos del tipo *String*

La estructura `str_array_t` contiene un puntero al arreglo identificado como *data* y la cantidad de elementos ocupados como *size*.



```
typedef struct str_array {
uint8_t size;
uint8_t capacity;
char** data;
} str_array_t;
```

Además como restricción para la entrega recuerden modificar solamente el archivo denominado `ej1.asm`, es decir su solución debe estar contenida por completo en este archivo.

Tema 2

Implementar las siguientes funciones en asm:

- `str_array_t* strArrayNew(uint8_t capacity)`
  Crea un array de strings nuevo con capacidad indicada por `capacity`.
- `uint8_t strArrayGetSize(str_array_t* a)`
  Obtiene la cantidad de elementos ocupados del arreglo.
- `char* strArrayGet(str_array_t* a, uint8_t i)`
  Obtiene el i-esimo elemento del arreglo, si `i` se encuentra fuera de rango, retorna 0.
- `char* strArrayRemove(str_array_t* a, uint8_t i)`
  Quita el i-esimo elemento del arreglo, si `i` se encuentra fuera de rango, retorna 0. El arreglo es reacomodado de forma que ese elemento indicado sea quitado y retornado.
- `void strArrayDelete(str_array_t* a)`
  Borra el arreglo, para esto borra todos los strings que el arreglo contenga, junto con las estructuras propias del tipo arreglo.

Estructura `Array`

Además la estructura cuenta con la siguiente función:

- `void strArrayPrint(str_array_t* a, FILE* pFile)`
  Escribe en el *stream* indicado por `pFile` el arreglo almacenado en `a`. Para cada string llama a la función de impresión strPrint. El formato del arreglo será: $[x_0, \ldots, x_{n-1}]$, suponiendo que x_i es el resultado de escribir el i-ésimo elemento.