CC3301: Programación de Software de Sistemas, Primavera 2009
Rodrigo Dueñas F. - Fabián Bravo A.

Auxiliar 1

Miércoles, 12 de Agosto, 2009

Problema 1.1: Jugando con strings

Programe la función `int strlen(char* u)` que devuelve el largo de un string.

Problema 1.2: Jugando con strings

Programe la función `char* strconcat(char *u, char *v)` que devuelve un string con ambos strings concatenados. Para ello sólo utilice malloc y la función programada anteriormente.

Problema 1.3: Jugando con strings

Explique qué hace este código (suponga que `buf` es un arreglo de chars):

```
char *p = buf;
while(*p){
  *p = *(p+1);
  p++;
}
```

Problema 2: Manejo de entrada y salida estándar

Utilizando las funciones del **problema 1** escribe un programa en C que a toda línea ingresada por la entrada estándar se le agrega como prefijo la frase 'CC31A dice: '. Por ejemplo:

```
$> ./prefijo
Hola
CC31A dice: Hola
Chao
CC31A dice: Chao
Prueba
CC31A dice: Prueba
```

Para leer y escribir de la entrada estándar puede utilizar las funciones de la librería `stdio.h`

- `char *fgets(char *s, int n, FILE *stream)`
- `int fputs(const char *s, FILE *stream)`

La entrada estándar es `stdin` y la salida estándar es `stdout` y se deben pasar como `stream`

Problema 3: Estructuras de control, malloc y free

Revise el siguiente código y explique porqué no está bien escrito.

```
int main() {
   int i = -1;
   int j;

   char **p = (char **)malloc(sizeof(char *)*2048);

   while(++i < 2048) {
      p[i] = (char *)malloc(1024*1024 + 1);

      for(j = 0; j < 1024*1024; ++j)
         *( *(p + i) +j) = 'b';

      *(p[i] + 1024*1024) = '\0';
   }

   return EXIT_SUCCESS;
}
```

Problema 4: Structs

Implementar una estructura de datos que simule una caja que guarde copias de una variable del tipo value (que tiene las mismas características que un entero). La caja posee además de una variable del tipo value un contador para ver cuantas copias posee de la variable. Las operaciones que realizables sobre una caja son:

- `void setValue(Box *b, value x)`

  Guarda una variable *value* en la caja, la cantidad inicial de copias es 1.

- `void setCounter(Box *b, int count)`

  Define cuantas copias del *value* posee la caja.

- *`value getCopy(Box b)`

  Retorna una de las copias de la caja, si no quedan copias retorna -1.

- *`void printBox(Box b)`

  Imprime el valor de la caja y la cantidad de copias que quedan.