Universidad de Chile
Departamento de Ciencias de la Computación
CC3301 - Programación de Software de Sistemas

Auxiliar 4 - Estructuras y Memoria

Profesor: Luis Mateu
Auxiliares: Gerard Cathalifaud
Vicente González
Joaquín López
Rodrigo Urrea

Resumen

Operador *

El operador * es para acceder al contenido de un puntero(lo que apunta). por ejemplo:

```
        int a = 5;
        int *pa = &a;
        printf("%d\n", *pa);
```

Debería de printear 5. También podemos cambiar el contenido de lo que apunta el puntero y por lo tanto cambiando el valor de la variable. Continuando el ejemplo

```
        *pa = 10;
        printf("%d\n" , a);
```

Debería de printear 10. La utilidad de esto es que, si nosotros queremos que una función cambie un parámetro en algún punto del código, le podemos dar un puntero con su dirección y en algún momento desreferenciarlo para cambiarlo.

Malloc and free

Para usar las funciones para alocar memoria necesitamos hacer un `#include <stdlib.h>`. Recuerden que siempre que alloquemos memoria hay que liberarla! La sintaxis típica para usarlo es si quiero pedir digamos 10 espacios para unos `int` y despues liberarlo es:

```
    int *espacios = malloc(sizeof(int)*10);
    ...//codigo entremedio
    free(espacios)
```

Structs

Podemos definir estructuras con la siguiente sintaxis

```
    typedef struct s {
        t1 campo1;
        t2 campo2;
        ...
    } MyStruct
    MyStruct myst;
```

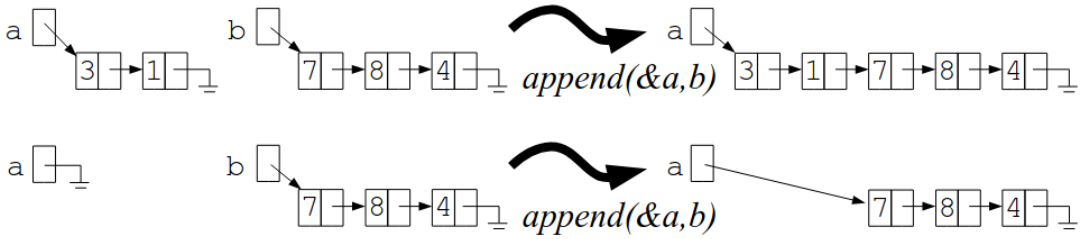
y podemos hacer un typedef para evitar tener que escribir `struct s` y solo escribir `MyStruct`. Podemos acceder a los cambios via la notacion punto `t1 xd = myst.campo1;`

Problemas

P1 Programe la función append que adjunta la lista enlazada en b a la lista *pa. El encabezado de la función es la siguiente.

```
typedef struct nodo {
    int x;
    struct nodo *sgte;
} Nodo;
void append(Nodo **pa, Nodo *b);
```

No puede usar malloc! Reutilicen los Nodos que reciben (ie *pa y b). El resultado debe de conservar el orden como se puede observar en la siguiente figura



P2 Implemente una cola FIFO que permita hacer las siguientes operaciones:

```
Cola *creaCola();
void put(Cola* q, char *str);
char *get(Cola* q);
void freeCola(Cola *q);
```

Las operaciones principales (`put` y `get`) se deben ejecutar en tiempo $O(1)$. Use una lista enlazada.

P3 (Propuesto) Implemente, Basándose en la P2, una pila LIFO que permita las siguientes operaciones

```
Pila *creaPila();
void push(Pila* s, char *str);
char *pop(Pila* s);
void freePila(Pila *s);
```