Universidad Nacional Autónoma de México
Semestre 2020-2
Compiladores
Generación de código intermedio
y Comprobación de tipos
Profesor: Adrián Ulises Mercado Martínez
Programa 4

1. Implementar una estructura de datos en la que se puedan guardar de forma dinámica las cuadruplas de código intermedio que se van generando.
2. Implementar las funciones auxiliares para realizar backpatch.
3. El código debe estar separado en archivos .h y .c
4. En main.c escribir código donde se hagan pruebas con datos para crear una pila insertar dos tablas llenar con tres elementos la tabla de la cima e imprimirla, sacar de la pila y liberar la memora de la pila, tanto para los tipos como para los símbolos.
5. Cada función tiene que estar documentada con la fecha y persona que la programo. Además de una muy breve descripción de lo que hace la función.
6. En caso de hacer cambios en alguna función agregar fecha de modificación y nombre de quién la modifico.

Se pueden utilizar como guía el siguiente código

cuadruplas.h

```
typedef struct cuad QUAD;

struct cuad{
  char *op;
  char *arg1;
  char *arg2;
  char *res;
  CUAD *next;
};

typedef struct code{
  CUAD *head;
  CUAD *tail;
} CODE;


CUAD *init_quad(); //Reserva memoria para una cuadrupla
void finish_quad(QUAD *c); // Libera la memoria de una cuadrupla

CODE *init_code(); //Reserva memoria para el codigo
void finish_code(CODE *c);// Libera la memoria de la lista ligada del codigo

void append_quad(CODE* C, CUAD *cd);// Agrega una cuadrupla al final de la lista
```

backpatch.h

```
#include "cuadruplas.h"

typedef struct index INDEX;

struct index{
  char *indice;
  INDEX *next;
};

typedef struct list_index{
  INDEX *head;
```

```
  INDEX *tail;
} LINDEX;


INDEX *init_index(); //Reserva memoria para un nodo de indice
void finish_index(INDEX *i); // Libera la memoria de un nodo indice

LINDEX *init_list_index(INDEX *i); //Reserva memoria para la lista de indices e inserta el primero
void finish_list_index(CODE *c);// Libera la memoria de la lista y de todos los nodos dentro

void append_index(LINDEX* l, INDEX *i);// Agrega un nodo indice al final de la lista

LINDEX *combinar(LINDEX l1, LINDEX l2)// retorna una lista ligada de la concatenacion de l1 con l2
void backpatch(CODE *c, LINDEX l, char *label);// Reemplaza label en cada aparicion de un indice en la
    cuadruplas del codigo c
```

tipos.h

```
#include "cuadruplas.h"

/*
 * Recibe dos tipos, cada uno es su indice hacia la tabla de tipos en la cima de la pila
 * Si el primero es mas chico que el segundo  genera la cuadrupla
 * para realizar la ampliacion del tipo.
 * Retorna la nueva direccion generada por la ampliacion
 * o la direccion original en caso de que no se realice
 */
char *ampliar(char *dir, int t1, int t2, CODE *c);

/*
 *Recibe dos tipos, cada uno es su indice hacia la tabla de tipos en la cima de la pila
 * Si el primero es mas grande que el segundo  genera la cuadrupla
 * para realizar la reduccion del tipo.
 * Retorna la nueva direccion generada por la reduccion
 * o la direccion original en caso de que no se realice
 */
char *reducir(); //Reserva memoria para un nodo de indice

/*
 * Recibe dos tipos, cada uno es indice hacia la tabla de tipos en la cima de la pila
 * retorna el indice de mayor dimension
 */
int max(int t1, int t2);


/*
* Recibe dos tipos, cada uno es indice hacia la tabla de tipos en la cima de la pila
* retorna el indice de menor dimension
*/
int min(int t1, int t2);
```