增删改查 … 记核心 后按框架来 → 封装 核心 · 周围 声明

(C (.h test 同)

.h　见下　入数合法性 空满影响 return free　见左

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "sqlist.h"

sqlink list_create() {
      //malloc
      sqlink L;

      L =(sqlink)malloc(sizeof(sqlist));
      if (L == NULL) {
            printf("list malloc failed\n");
            return L;
      }

      //initialize
      memset(L, 0, sizeof(sqlist));
      L->last = -1;

      //return
      return L;
}
```

malloc → size =() 判

初 → data = 0 → memset
last = -1 → L->last.

```
/*
 * @ret   0-success   -1-failed
 * */
int list_clear(sqlink L) {
      if (L == NULL)
            return -1;

      memset(L, 0, sizeof(sqlist));
      L->last = -1;

      return 0;
}
```

data = 0 → memset.
last = -1 → L->last.

```
int list_free(sqlink L){
      if (L == NULL)
            return -1;
      free(L);
      L = NULL;
      return 0;
}
```

free → P
P = NULL

```
/*
 * list_empty: Is list empty?
 * para L: list
 * @ret  1--empty   0--not empty
 * */
int list_empty(sqlink L) {
      if (L->last == -1)
            return 1;
      else
            return 0;
}
```

last -1 1
0 0

```
int list_length(sqlink L) {
      if (L == NULL)
            return -1;
```

返 Last +1

```
	return (L->last+1);
}

/*
 * @ret  -1--not exist   pos
 * */
int list_locate(sqlink L, data_t value) {
	int i ;
	for (i = 0; i <= L->last; i++) {
		if (L->data[i] == value)
			return i;
	}

	return -1;
}
```

遍历,找到的话,返i

```
int list_insert(sqlink L, data_t value, int pos) {
	int i;

	//full
	if (L->last == N-1) {
		printf("list is full\n");
		return -1;
	}

	//check para    0<=pos<=Last+1   [0, last+1]
	if (pos < 0 || pos > L->last+1) {
		printf("Pos is invalid\n");
		return -1;
	}

	//move
	for (i = L->last; i >= pos; i--) {
		L->data[i+1] = L->data[i];
	}

	//update value last
	L->data[pos] = value;
	L->last++;

	return 0;
}
```

腾 → last pos --
+1 = i

放 → pos = value

改址 → (last)

```
int list_show(sqlink L) {
	int i;

	if (L == NULL)
		return -1;
	if (L->last == -1)
		printf("list is empty\n");

	for (i = 0; i <= L->last; i++) {
		printf("%d ", L->data[i]);
	}
	puts("");

	return 0;
}
```

遍历. 打印

```c
int list_delete(sqlink L, int pos) {
	int i;

	if (L->last == -1) {
		printf("list is empty\n");
		return -1;
	}

	//pos [0, last]
	if (pos < 0 || pos > L->last) {
		printf("delete pos is invalid\n");
		return -1;
	}

	//move  [pos+1, last]
	for (i = pos+1; i <= L->last; i++) {
		L->data[i-1] = L->data[i];
	}

	//update
	L->last--;

	return 0;
}

int list_merge(sqlink L1, sqlink L2) {
	int i = 0;
	int ret;

	while (i <= L2->last){
		ret = list_locate(L1, L2->data[i]);
		if (ret == -1) {
			if (list_insert(L1, L2->data[i], L1->last+1) == -1)
				return -1;
		}

		i++;
	}
	return 0;
}

int list_purge(sqlink L) {
	int i;
	int j;

	if (L->last == 0)
		return 0;

	i = 1;
	while (i <= L->last) {
		j = i-1;
		while (j >= 0) {
			if (L->data[i] == L->data[j]) {
				list_delete(L, i);
				break;
			} else {
				j--;
			}
		}
```

腾 → i=pos+1 <=last i++
i-1 i

改地 → Last

遍L2

如不在

就入

(個一個要考慮有必要考慮成功不)

$|\longrightarrow$

$\longleftarrow \cdot$ == 删
!= 继加

(while break 条件实现)

```
            if ( j < 0) {
                  i++;
            }
      }

      return 0;
}
```