

The Josephus Puzzle Revisited

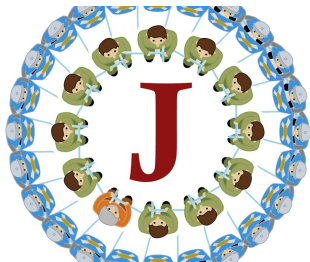
— Struct, Linked List, and Function Pointer

魏恒峰

hfwei@nju.edu.cn

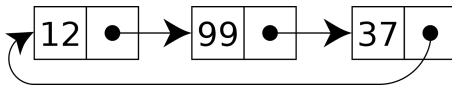
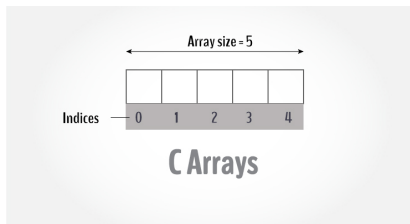
2017 年 11 月 10 日





The Josephus Puzzle

$$J(n) = ?$$



```
struct _node {...};  
struct _linkedlist {...};
```

```
struct _node {...};  
struct _linkedlist {...};
```

```
scanf("%d", &n);
```

```
LinkedList list;  
initialize_list(&list);
```

```
sit_in_circle(&list, n);
```

```
kill_until_one(&list);
```

```
show(&list);
```

Struct

(struct-point.c)

```
typedef struct _point {  
    int x;  
    int y;  
} Point;
```

```
typedef struct _point {  
    int x;  
    int y;  
} Point;
```

```
struct _point p1 = {1, 1};  
p1.x = 11;
```

```
Point p2 = {.x = 2, .y = 2};
```

```
Point ps[5];
```

```
Point *p = &p2;  
p->x = 22;
```



```
typedef struct _point {  
    char c;  
    int x;  
    int y;  
} Point;
```

```
Point p = {.c = 'o', .x = 0, .y = 0};  
sizeof(p);
```

```
typedef struct _point {  
    int x;  
    int y;  
} Point;
```

```
void show(Point p);
```

```
void update(Point *p, int x, int y);
```

```
Point add(Point p1, Point p2);
```

```
typedef struct _point {  
    int x;  
    int y;  
} Point;
```

```
typedef struct _rect {  
    Point lup;  
    Point rlp;  
} Rect;  
  
Rect r;  
  
r.lup.x = 1;
```

Linked List

(`linkedlist.h` `linkedlist-test.c`)

```
typedef struct _node {  
    void *data;  
    struct _node *next;  
} Node;
```

```
typedef struct _linkedlist {  
    Node *head;  
    Node *tail;  
} LinkedList;
```

```
void initialize_list(LinkedList *list);

int is_empty(LinkedList *list);
int is_singleton(LinkedList *list);

void add_tail(LinkedList *list, void *data);

void delete_next(LinkedList *list, Node *pre);
```

```
void kill_until_one(LinkedList *list) {  
    Node *tmp = list->head;  
  
    while (! is_singleton(list)) {  
        delete_next(list, tmp);  
        tmp = tmp->next;  
    }  
}
```

(josephus-linkedlist.c)

```
void delete_node(LinkedList *list, Node *node);  
  
void insert(LinkedList *list, Node *pre);
```


Function Pointer

```
int (*fptr)(int); // fptr is a function pointer

int square(int num) {
    return num * num;
}
```

```
int n = 5;
fptr = square; // fptr points to a function
fptr(n);
```

```
typedef void (*fptr_show)(void *data);  
  
void show(LinkedList *list, fptr_show show);  
  
(linkedlist.h)
```

```
typedef void (*fptr_show)(void *data);  
  
void show(LinkedList *list, fptr_show show);  
  
(linkedlist.h)
```

```
void show_integer(const int *integer);  
  
show(&list, show_integer);  
  
(josephus-linkedlist.c)
```

```
typedef int (*fptr_compare)(void *data1, void *  
    data2);
```

```
Node *get_node(LinkedList *list, fptr_compare  
    compare, void *data);
```

([linkedlist.h](#))

```
typedef int (*fptr_compare)(void *data1, void *  
    data2);
```

```
Node *get_node(LinkedList *list, fptr_compare  
    compare, void *data);
```

(linkedlist.h)

```
int compare_integer(int *data1, int *data2);
```

```
get_node(&list, compare_integer, &val);
```

(josephus-linkedlist.c)

```
void qsort (void *base, size_t num, size_t size,  
            int (*compar)(const void*, const void*));  
  
#include <stdlib.h>
```

Thank
You!