**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김 영 재 교수님 |
| 학번 : | 20210041 |
| 이름 : | 박 민 성 |
|  |  |

1. **Additional Implementation**

**main.c**

|  |  |
| --- | --- |
| **Prototype** | int string\_split(char \*str, char argv[10][50]); |
| **Parameter** | 원본 문자열 \*str, 공백 단위로 분리할 문자열 배열 argv |
| **Return** | 총 command의 수 리턴 |
| **Function** | 용이한 command 처리를 위해 main()에서 fgets()를 이용해 입력 받은 문자열을 공백 단위로 나누는 작업 수행 |

|  |  |
| --- | --- |
| **Prototype** | int main() |
| **Parameter** |  |
| **Return** |  |
| **Function** | - using variables:  char input, argv, int argc (for input data)  struct main\_list, main\_hash, main\_bitmap (각각의 자료구조 생성 후 명령에 따라 관리)  - string\_split()을 이용해 명령을 단어 단위로 관리하면서 처리 |

**list.c**

|  |  |
| --- | --- |
| **Prototype** | bool less (const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | 크기 비교를 위한 list\_elem 변수 \*a, \*b |
| **Return** | a보다 b의 데이터 값이 더 클 경우 true. 나머지 경우에 false |
| **Function** | list\_entry 매크로를 활용해서 두 변수 a, b의 크기 비교 |

|  |  |
| --- | --- |
| **Prototype** | void create\_list (struct list \*\*List, char \*list\_name) |
| **Parameter** | list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 list name을 토대로 index에 리스트 생성 |

|  |  |
| --- | --- |
| **Prototype** | void delete\_list (struct list \*\*List, char \*list\_name) |
| **Parameter** | list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 list name을 토대로 해당 index의 리스트 제거 |

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_list (struct list \*\*List, char \*list\_name) |
| **Parameter** | list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 list name을 토대로 해당 index의 리스트에 저장되어 있는 값들 출력(값이 없을 경우 출력하지 않음) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \* find\_elem (struct list \*list, int position) |
| **Parameter** | 탐색하고 싶은 list, list에서 몇 번째 위치에 있는지 나타내는 변수 position |
| **Return** | list에서 head 다음 원소부터 position번 째에 있는 원소 반환 |
| **Function** | - 원소 삽입과 제거 등의 작업을 수행할 때 리스트를 index 기반으로 접근하는 것을 도와주는 함수  - 반복문을 통해 list\_begin()부터 position번 리스트 순회 후, 해당 위치에 있는 원소 반환 |

|  |  |
| --- | --- |
| **Prototype** | void insert\_elem\_list (struct list \*\*List, char \*list\_name, int position, int value) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name  - 추가할 위치 position, 추가할 값 value |
| **Return** | void |
| **Function** | - list\_push\_back(), list\_push\_front()와 find\_elem() 함수를 통해 원하는 위치에 값 삽입  - position이 -1이면 list\_push\_back() |

|  |  |
| --- | --- |
| **Prototype** | void print\_list (struct list \*\*List, char \*list\_name, int position) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name  - 원하는 위치 position(0: front/ -1: back) |
| **Return** | void |
| **Function** | list의 front와 back에 위치한 원소를 list\_back()과 list\_begin()을 이용해서 출력 |

|  |  |
| --- | --- |
| **Prototype** | void ordered\_insert\_elem\_list (struct list \*\*List, char \*list\_name, int value) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name  - 추가할 값 value |
| **Return** | void |
| **Function** | - list\_entry()를 이용해 추가할 값을 가지고 있는 새로운 원소를 생성  - list\_insert\_ordered()를 이용해 오름차순을 유지하며 원소 삽입 |

|  |  |
| --- | --- |
| **Prototype** | void remove\_elem\_list (struct list \*\*List, char \*list\_name, int position) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name  - 제거할 위치 position(0: front/ -1: back / otherwise) |
| **Return** | void |
| **Function** | list\_pop\_back(), list\_pop\_front()와 find\_elem() 함수를 통해 원하는 위치의 값 제거 |

|  |  |
| --- | --- |
| **Prototype** | void print\_max\_elem\_list (struct list \*\*List, char \*list\_name) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name |
| **Return** | void |
| **Function** | list\_max()를 이용해 list에서 가장 큰 값을 가지고 있는 원소 탐색 후 list\_entry()를 이용해 출력 |

|  |  |
| --- | --- |
| **Prototype** | void print\_min\_elem\_list (struct list \*\*List, char \*list\_name) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name |
| **Return** | void |
| **Function** | list\_min()를 이용해 list에서 가장 작은 값을 가지고 있는 원소 탐색 후 list\_entry()를 이용해 출력 |

|  |  |
| --- | --- |
| **Prototype** | void shuffle\_list (struct list \*\*List, char \*list\_name) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name |
| **Return** | void |
| **Function** | - list\_shuffle()을 이용하기 위해 호출하는 함수  - list\_shuffle()을 호출하기 전에 사용할 list의 index와 list 내부 확인 |

|  |  |
| --- | --- |
| **Prototype** | void splice\_list (struct list \*a, struct list \*b, int index, int start, int end) |
| **Parameter** | list a, a에 잘라서 붙일 list b, a에 붙이기 시작할 위치 index, b에서 자를 범위를 나타내는 start, end |
| **Return** | void |
| **Function** | list\_splice(before, first, last) 안에 들어갈 원소들을 find\_elem()으로 탐색한 후 list\_splice 실행 |

|  |  |
| --- | --- |
| **Prototype** | void swap\_list (struct list \*\*List, char \*list\_name, int position1, int position2) |
| **Parameter** | - list들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 list\_name  - 교환할 위치 position1, position2 |
| **Return** | void |
| **Function** | list에서 position1, position2에 해당하는 위치에 있는 원소를 find\_elem()으로 찾은 후, list\_swap()을 통해 두 원소 교환 |

**hash.c**

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*new\_elem (int value) |
| **Parameter** | 추가할 값 value |
| **Return** | 새로운 hash elem\* 리턴 |
| **Function** | - 추가할 값을 원소에 저장 후 해당 원소 포인터를 반환  - Hash 테이블에 값을 쉽게 저장할 수 있게 함 |
| **Prototype** | unsigned hash\_func (const struct hash\_elem \*e, void \*aux) |
| **Parameter** | 해싱하기 전 원소 e |
| **Return** | 해싱한 후의 해시값 반환 |
| **Function** | 해싱 함수를 선택해서 해싱을 제공하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_print (struct hash\_elem \*e, void \*aux) |
| **Parameter** | 출력할 원소 e |
| **Return** | void |
| **Function** | 해시 테이블 안의 원소 e가 가지고 있는 값 출력 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square (struct hash\_elem \*e, void \*aux) |
| **Parameter** | 제곱할 원소 e |
| **Return** | void |
| **Function** | 원소 e의 값을 제곱하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple (struct hash\_elem \*e, void \*aux) |
| **Parameter** | 세제곱할 원소 e |
| **Return** | void |
| **Function** | 원소 e의 값을 세제곱하는 함수 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destruct (struct hash\_elem \*e, void \*aux) |
| **Parameter** | 소멸시킬 원소 e |
| **Return** | void |
| **Function** | 원소 e를 소멸 |

|  |  |
| --- | --- |
| **Prototype** | void create\_hash (struct hash \*\*Hash, char \*hash\_name) |
| **Parameter** | hash들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 hash\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 hash name을 토대로 index에 hash 생성 |

|  |  |
| --- | --- |
| **Prototype** | void delete\_hash (struct hash \*\*Hash, char \*hash\_name) |
| **Parameter** | hash들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 hash\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 hash name을 토대로 index에 hash 제거 |

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_hash (struct hash \*\*Hash, char \*hash\_name) |
| **Parameter** | hash들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 hash\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 hash name을 토대로 해당 index의 hash에 저장되어 있는 값들 출력(값이 없을 경우 출력하지 않음) |

|  |  |
| --- | --- |
| **Prototype** | void apply\_hash (struct hash \*\*Hash, char \*hash\_name, int command) |
| **Parameter** | - hash들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 hash\_name  - 명령의 종류 command(0: double, 1: triple) |
| **Return** | void |
| **Function** | main()에서 전달받은 hash name을 토대로 해당 index의 hash에 연산 수행 |

**bitmap.c**

|  |  |
| --- | --- |
| **Prototype** | void dumpdata\_bitmap (struct bitmap \*\*Bitmap, char \*bitmap\_name) |
| **Parameter** | bitmap들이 저장되어 있는 구조체 배열, 사용할 구조체의 이름이 있는 문자열 bitmap\_name |
| **Return** | void |
| **Function** | main()에서 전달받은 bitmap name을 토대로 해당 index의 bitmap에 저장되어 있는 값들 출력(값이 없을 경우 출력하지 않음) |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void list\_swap (struct list\_elem \*a, struct list\_elem \*b) |
| **Parameter** | list\_elem a, b 두 변수를 입력받아 서로 위치를 교환 |
| **Return** | void |
| **Function** | - a와 b가 서로 인접하지 않은 경우  - a와 b가 서로 인접한 경우(b, a순서의 경우 순서를 바꿔서 다시 호출)  위의 두 경우로 나누어서 직접적인 데이터 교환 없이 리스트에서 두 변수의 위치를 교환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list) |
| **Parameter** | list 구조체를 입력받아 내부 변수들의 순서를 랜덤하게 섞음 |
| **Return** | void |
| **Function** | 피셔-예이츠 셔플(Fisher-Yates shuffle)을 응용해서 사용  -교환의 용이성을 위해 변수 array\_list를 이용해 list에 index 부여  - 인덱스를 하나씩 줄여 가며 난수를 활용해 범위 안(0~i-1)에 있는 변수와 인덱스가 가리키는 변수를 서로 교환  - Time complexity: O(N) |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int i) |
| **Parameter** | Hashing을 위한 정수 i |
| **Return** | Hashing이 완료된 unsigned h 리턴 |
| **Function** | - Hash 값을 고르게 분포시키기 위해 xor연산과 shift연산을 활용해 비트를 섞음  - 소수를 곱하는 곱셈 hashing을 적용해 hash 값의 균등한 분포를 유도하고 충돌을 방지 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \* bitmap\_expand (struct bitmap \*bitmap, size\_t size) |
| **Parameter** | 크기를 확장시킬 bitmap, 확장시킬 크기 size |
| **Return** | 확장에 성공하면 확장한 bitmap, 확장에 실패하면 NULL 리턴 |
| **Function** | - bitmap\_create 함수를 이용해 bitmap의 크기 확장  - 원본 bitmap을 확인하며 원본 bitmap의 값을 새로운 bitmap에 복사  - bitmap\_test()를 이용해 해당 위치의 값이 true라면 새로운 bitmap에 bitmap\_mark() 사용  - Time complexity: O(N) |