**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 김영재 |
| 학번 : | 20231609 |
| 이름 : | 정희선 |
|  |  |

1. **Additional Implementation**

**\* list**

|  |  |
| --- | --- |
| **Prototype** | struct list\_item { struct list\_elem elem; int data; }; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 정수 데이터를 저장할 수 있는 리스트 요소 구조체 |

|  |  |
| --- | --- |
| **Prototype** | void delete\_list\_elem(struct list\_elem \*e); |
| **Parameter** | struct list\_elem \*e: 삭제할 리스트 요소 |
| **Return** | 없음 |
| **Function** | 리스트 요소와 연결된 list\_item 구조체를 메모리에서 해제 |

|  |  |
| --- | --- |
| **Prototype** | bool my\_list\_compare(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux); |
| **Parameter** | - const struct list\_elem \*a: 비교할 첫 번째 요소  - const struct list\_elem \*b: 비교할 두 번째 요소  - void \*aux: 보조 데이터 (이 함수에서는 사용되지 않음) |
| **Return** | a의 데이터 값이 b보다 작으면 true, 그렇지 않으면 false |
| **Function** | 두 리스트 요소의 정수 데이터 값을 비교 |

**\* hash**

|  |  |
| --- | --- |
| **Prototype** | struct hash\_item { struct hash\_elem elem; int data; }; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 정수 데이터를 저장할 수 있는 해시 테이블 요소 구조체 |

|  |  |
| --- | --- |
| **Prototype** | unsigned my\_hash\_func (const struct hash\_elem \*e, void \*aux); |
| **Parameter** | - const struct hash\_elem \*e: 해시할 요소  - void \*aux: 보조 데이터 (이 함수에서는 사용되지 않음) |
| **Return** | 해시된 정수 값 |
| **Function** | hash\_elem에서 hash\_item으로 변환하여 데이터 값을 해시 |

|  |  |
| --- | --- |
| **Prototype** | bool my\_hash\_compare(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux); |
| **Parameter** | - const struct hash\_elem \*a: 비교할 첫 번째 요소  - const struct hash\_elem \*b: 비교할 두 번째 요소  - void \*aux: 보조 데이터 (이 함수에서는 사용되지 않음) |
| **Return** | a의 데이터 값이 b보다 작으면 true, 그렇지 않으면 false |
| **Function** | 두 해시 요소의 정수 데이터 값을 비교 |

|  |  |
| --- | --- |
| **Prototype** | void delete\_hash\_elem (struct hash\_elem \*e, void \*aux); |
| **Parameter** | - struct hash\_elem \*e: 삭제할 해시 요소  - void \*aux: 보조 데이터 (이 함수에서는 사용되지 않음) |
| **Return** | 없음 |
| **Function** | 해시 요소와 연결된 hash\_item 구조체를 메모리에서 해제 |

|  |  |
| --- | --- |
| **Prototype** | void triple(struct hash\_elem \*e, void \*aux); |
| **Parameter** | - struct hash\_elem \*e: 수정할 해시 요소  - void \*aux: 보조 데이터 (이 함수에서는 사용되지 않음) |
| **Return** | 없음 |
| **Function** | 해시 요소의 데이터 값을 세제곱으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | void square(struct hash\_elem \*e, void \*aux); |
| **Parameter** | - struct hash\_elem \*e: 수정할 해시 요소  - void \*aux: 보조 데이터 (이 함수에서는 사용되지 않음) |
| **Return** | 없음 |
| **Function** | 해시 요소의 데이터 값을 제곱으로 변경 |

**\* bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct hash\_item { struct hash\_elem elem; int data; }; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 정수 데이터를 저장할 수 있는 해시 테이블 요소 구조체 |

|  |  |
| --- | --- |
| **Prototype** | bool string\_to\_bool(const char \*str); |
| **Parameter** | const char \*str: 변환할 문자열 |
| **Return** | 문자열이 "true"이면 true, 그렇지 않으면 false |
| **Function** | 문자열 "true"를 boolean true로 변환, 그 외는 false |

**\* main**

|  |  |
| --- | --- |
| **Prototype** | #define MAX\_NUM 10  #define MAX\_ARGS 10  #define MAX\_BITMAP\_SIZE 64 |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | - 생성 가능한 최대 데이터 구조 배열 크기 제한  - 명령어에서 파싱할 수 있는 최대 인자 수 제한  - 생성 가능한 최대 비트맵 크기 제한 |

|  |  |
| --- | --- |
| **Prototype** | typedef enum {...} L\_command;  typedef enum {...} H\_command;  typedef enum {...} B\_command; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | - 리스트 관련 명령어를 식별하는 열거형  - 해시 테이블 관련 명령어를 식별하는 열거형  - 비트맵 관련 명령어를 식별하는 열거형 |

|  |  |
| --- | --- |
| **Prototype** | struct list\* ListArr[MAX\_NUM];  struct hash\* HashArr[MAX\_NUM];  struct bitmap\* BitmapArr[MAX\_NUM]; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | - 생성된 리스트 구조체들의 포인터를 저장하는 배열  - 생성된 해시 테이블 구조체들의 포인터를 저장하는 배열  - 생성된 비트맵 구조체들의 포인터를 저장하는 배열 |

|  |  |
| --- | --- |
| **Prototype** | int ListNum=0, HashNum=0, BitmapNum=0; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 각 자료구조 타입별 현재 생성된 개수를 추적하는 변수들 |

|  |  |
| --- | --- |
| **Prototype** | L\_command get\_list\_cmd(const char\* cmd);  H\_command get\_hash\_cmd(const char\* cmd);  B\_command get\_bitmap\_cmd(const char\* cmd); |
| **Parameter** | const char\* cmd: 변환할 문자열 명령어 |
| **Return** | - 일치하는 L\_command 열거형 값, 일치하는 명령어가 없으면 0 반환  - 일치하는 H\_command 열거형 값, 일치하는 명령어가 없으면 0 반환  - 일치하는 B\_command 열거형 값, 일치하는 명령어가 없으면 0 반환 |
| **Function** | - 문자열 명령어를 리스트 명령어 열거형으로 변환  - 문자열 명령어를 해시 테이블 명령어 열거형으로 변환  - 문자열 명령어를 비트맵 명령어 열거형으로 변환 |

|  |  |
| --- | --- |
| **Prototype** | void process\_list\_cmd(L\_command cmd, char\*\* args);  void process\_hash\_cmd(H\_command cmd, char\*\* args);  void process\_bitmap\_cmd(B\_command cmd, char\*\* args); |
| **Parameter** | - L\_command/H\_command/B\_command cmd: 실행할 리스트/해시 테이블/비트맵 명령어 열거형  - char\*\* args: 명령어와 함께 전달된 인자 배열 |
| **Return** | 없음 |
| **Function** | - 리스트 명령어를 처리하고 적절한 리스트 연산 수행  - 해시 테이블 명령어를 처리하고 적절한 해시 연산 수행  - 비트맵 명령어를 처리하고 적절한 비트맵 연산 수행 |

|  |  |
| --- | --- |
| **Prototype** | void process\_command(char \*command); |
| **Parameter** | char \*command: 사용자가 입력한 원본 명령어 문자열 |
| **Return** | 없음 |
| **Function** | 사용자 입력 명령어를 파싱하고 적절한 처리 함수로 전달 |

|  |  |
| --- | --- |
| **Prototype** | int main(); |
| **Parameter** | 없음 |
| **Return** | 프로그램 종료 코드 (0: 정상 종료) |
| **Function** | 프로그램의 메인 함수, 사용자 입력을 읽고 처리하는 루프 실행  <주요 처리 로직>  - process\_command 함수는 입력된 명령어를 공백으로 분리하여 args 배열에 저장하고, 첫 번째 토큰을 기준으로 다음과 같은 기본 명령어를 처리한다:   1. create: 새로운 자료구조(list, hashtable, bitmap) 생성 2. delete: 이름으로 지정된 자료구조 삭제 3. dumpdata: 자료구조의 모든 데이터를 화면에 출력 4. quit: 프로그램 종료   - 기본 명령어가 아닌 경우, 각 자료구조별 특화 명령어를 처리하는 함수(process\_list\_cmd, process\_hash\_cmd, process\_bitmap\_cmd)로 전달된다.  - 각 자료구조별 처리 함수는 해당 명령어에 맞는 자료구조 연산을 수행하고 필요한 경우 결과를 출력한다. |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | struct list {      struct list\_elem head;      struct list\_elem tail;      char\* name;    }; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 리스트에 이름을 부여할 수 있는 문자열 포인터 추가 |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem \*a, struct list\_elem \*b); |
| **Parameter** | - struct list\_elem \*a: 교환할 첫 번째 요소  - struct list\_elem \*b: 교환할 두 번째 요소 |
| **Return** | 없음 |
| **Function** | 리스트 내 두 요소의 위치를 교환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list \*list); |
| **Parameter** | struct list \*list: 섞을 리스트 |
| **Return** | 없음 |
| **Function** | 리스트 내 요소들을 무작위로 섞음  Fisher-Yates suffle algorithm 사용 |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | struct hash    {      size\_t elem\_cnt;      size\_t bucket\_cnt;      struct list \*buckets;      hash\_hash\_func \*hash;      hash\_less\_func \*less;      void \*aux;      char\* name;    }; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 해시 테이블에 이름을 부여할 수 있는 문자열 포인터 추가 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int i); |
| **Parameter** | int i: 해시할 정수 값 |
| **Return** | 해시된 unsigned 정수 값 |
| **Function** | 황금비(golden ratio)를 이용한 곱셈 해싱 방법으로 정수를 해시값으로 변환 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | struct bitmap  {      size\_t bit\_cnt;      elem\_type \*bits;      char\* name;  }; |
| **Parameter** | 없음 |
| **Return** | 없음 |
| **Function** | 비트맵에 이름을 부여할 수 있는 문자열 포인터 추가 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_expand(struct bitmap \*bitmap, int size); |
| **Parameter** | - struct bitmap \*bitmap: 확장할 원본 비트맵  - int size: 추가할 비트 수 |
| **Return** | 확장된 새 비트맵 포인터, 실패 시 NULL |
| **Function** | 기존 비트맵의 크기를 확장하여 새 비트맵 생성 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*b); |
| **Parameter** | - struct bitmap \*bitmap: 메모리를 해제할 비트맵 |
| **Return** | 없음 |
| **Function** | 기존 함수에서 비트맵의 name 필드 메모리 해제 부분 추가 구현 |