**Project #1 : MyLib**

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 |
| 학번 : | 20201610 |
| 이름 : | 이도안 |
|  |  |

1. **Additional Implementation**

**List, hash, bitmap과 연관되지 않은 Additional Implementation 없음**

1. **List**

**list\_init**

Prototype: void list\_init(struct list \*list)

Parameter: struct list \*list – 초기화할 리스트 구조체 변수의 주소.

Return: None.

Function: 파라미터로 받은 리스트를 초기화하여 빈 상태 ( head – tail ) 상태로 만들어준다.

**list\_begin**

Prototype: struct list\_elem \*list\_begin(struct list \*list)

Parameter: struct list \*list – 시작점을 찾을 리스트 변수

Return: 리스트의 시작점을 반환한다.

Function: 리스트의 시작점(head의 next)을 찾아 반환한다.

**list\_next**

Prototype: struct list\_elem \*list\_next(struct list\_elem \*elem)

Parameter: struct list\_elem \*elem – next element를 찾을 리스트 원소의 주소.

Return: 리스트 원소의 next element pointer.

Function: list element가 가리키는 다음 원소의 주소를 반환한다.

**list\_end**

Prototype: struct list\_elem \*list\_end(struct list \*list)

Parameter: struct list \*list – 끝을 찾을 리스트의 주소.

Return: 해당 리스트의 tail.

Function: 리스트의 tail값을 반환한다..

**list\_insert**

Prototype: void list\_insert(struct list\_elem \*before, struct list\_elem \*elem)

Parameters:

struct list\_elem \*before – 리스트의 원소가 삽입될 위치의 바로 다음 위치의 원소 pointer.

struct list\_elem \*elem – 삽입될 리스트 원소의 pointer.

Return: None.

Function: before원소가 위치하는 바로 전 위치에 해당 리스트 elem을 삽입한다.

**list\_remove**

Prototype: struct list\_elem \*list\_remove(struct list\_elem \*elem)

Parameter: struct list\_elem \*elem – 지울 원소 pointer

Return: 지운 element pointer.

Function: 지정 원소를 리스트에서 지우고 그 원소를 반환함.

**list\_push\_front**

Prototype: void list\_push\_front(struct list \*list, struct list\_elem \*elem)

Parameters:

struct list \*list – 원소를 push할 리스트의 포인터.

struct list\_elem \*elem – push할 새로운 원소.

Return: None.

Function: 리스트의 시작점에 원소를 삽입한다.

**list\_push\_back**

Prototype: void list\_push\_back(struct list \*list, struct list\_elem \*elem)

Parameters:

struct list \*list – push할 리스트의 포인터.

struct list\_elem \*elem – 리스트의 back에 삽입할 새로운 원소.

Return: None.

Function: 리스트의 back에 새로운 원소를 삽입한다.

**list\_pop\_front**

Prototype: struct list\_elem \*list\_pop\_front(struct list \*list)

Parameter: struct list \*list – pop할 list pointer.

Return: pop된 리스트의 원소를 반환.

Function: 리스트의 front에 있는 원소를 pop하고 반환한다.

**list\_pop\_back**

Prototype: struct list\_elem \*list\_pop\_back(struct list \*list)

Parameter: struct list \*list – pop할 list pointer.

Return: pop된 리스트의 원소를 반환.

Function: 리스트의 back에 있는 원소를 pop하고 반환한다.

**list\_size**

Prototype: size\_t list\_size(struct list \*list)

Parameter: struct list \*list – size를 찾을 리스트 포인터.

Return: 리스트의 원소의 개수.

Function: 리스트의 원소의 개수를 찾아 반환한다.

**list\_empty**

Prototype: bool list\_empty(struct list \*list)

Parameter: struct list \*list – 비었는지 확인할 리스트 포인터.

Return: 리스트가 빈 상태면 1, 아니면 0.

Function: 리스트가 head – tail 상태로 비었는지 확인하는 기능을 한다.

**list\_sort**

Prototype: void list\_sort(struct list \*list, list\_less\_func \*less, void \*aux)

Parameters:

struct list \*list – sort할 리스트의 포인터

list\_less\_func \*less – sort하는 데에 기준이 되는 비교함수

void \*aux – 비교함수에 parameter로 활용되는 추가 정보를 담는 변수.

Return: None.

Function: 파라미터로 제공된 비교함수와 추가변수에 따라서 리스트를 정렬해준다.

**list\_insert\_ordered**

Prototype: void list\_insert\_ordered(struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux)

Parameters:

struct list \*list – 원소를 삽입할 리스트의 포인터

struct list\_elem \*elem – 리스트에 삽입할 원소 포인터.

list\_less\_func \*less - 비교함수.

void \*aux – 비교함수에 사용될 추가 정보를 담는 변수.

Return: None.

Function: 비교함수를 사용하여 리스트의 정렬을 해치지 않도록 변수를 삽입하는 함수.

**list\_unique**

Prototype: void list\_unique(struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux)

Parameters:

struct list \*list - Pointer to the list.

struct list \*duplicates – unique 함수에 의해 복제된 함수가 담길 리스트 구조체 변수

list\_less\_func \*less – 비교함수

void \*aux – 비교함수에 이용될 추가 정보를 담는 변수

Return: None.

Function: 리스트에서 반복되는 element를 삭제하고, 선택적으로 다른 리스트에 저장한다.

**list\_max**

Prototype: struct list\_elem \*list\_max(struct list \*list, list\_less\_func \*less, void \*aux)

Parameters:

struct list \*list – max를 찾을 리스트의 포인터.

list\_less\_func \*less – 비교함수

void \*aux - 비교함수에 이용될 추가 정보를 담는 변수.

Return: 비교함수를 사용해 찾은 리스트에서 가장 큰 원소의 포인터

Function: 비교함수를 사용하여 리스트에서 가장 큰 원소를 찾고, 이를 반환한다.

**list\_min**

Prototype: struct list\_elem \*list\_min(struct list \*list, list\_less\_func \*less, void \*aux)

Parameters:

struct list \*list – max를 찾을 리스트의 포인터.

list\_less\_func \*less – 비교함수

void \*aux - 비교함수에 이용될 추가 정보를 담는 변수.

Return: 비교함수를 사용해 찾은 리스트에서 가장 작은 원소의 포인터

Function: 비교함수를 사용하여 리스트에서 가장 작은 원소를 찾고, 이를 반환한다.

**list\_swap**

Prototype: void list\_swap(struct list\_elem \*a, struct list\_elem \*b)

Parameters:

struct list\_elem \*a – swap할 첫 번째 원소

struct list\_elem \*b - swap할 두 번째 원소

Return: None.

Function: 리스트에서 두 원소를 swap한다. 단, 데이터값 뿐만이 아니라 주소 값을 포함하여 모든 정보를 swap한다.

**list\_shuffle**

Prototype: void list\_shuffle(struct list \*list)

Parameters:

struct list \*list - Pointer to the list.

Return: None.

Function: list의 element를 무작위로 섞는다. 이 함수에서는 두 번의 shuffle이 일어난다.

1. List의 앞 뒤에서 무작위로 pop하여 새로운 list에 push\_front한다.
2. 만들어진 새로운 리스트에서 다시 원래 리스트로 돌려주면서 무작위 위치에 삽입해준다.

이를 통해 shuffle했을 때 전의 원소 배치와 다른 상황이 나올 확률을 늘려주었다.

**data\_less**

Prototype: bool data\_less(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux)

Parameters:

const struct list\_elem \*a – 비교할 첫 번째 원소

const struct list\_elem \*b – 비교할 두 번째 원소

void \*aux – 비교함수에 사용될 추가 정보를 담는 변수

Return: 반환값 = (a < b)

Function: 원소 a와 b를 비교하여 a가 더 작으면 1을, b가 더 작으면 0을 반환한다.

1. **Hash Table**

**hash\_init**

Prototype: bool hash\_init(struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux)

Parameters:

struct hash \*h – 초기화할 해시 테이블.

hash\_hash\_func \*hash – 해시 값을 계산식을 가진 함수.

hash\_less\_func \*less – 해시 값 비교 함수.

void \*aux – 비교 함수에 전달될 추가 정보를 담은 변수.

Return: 초기화 성공 시 true, 실패 시 false.

Function: 해시 테이블을 초기화하고 해시 및 비교 함수를 설정한다.

**hash\_clear**

Prototype: void hash\_clear(struct hash \*h, hash\_action\_func \*destructor)

Parameters:

struct hash \*h – 원소를 제거할 해시 테이블.

hash\_action\_func \*destructor – 각 원소에 대해 호출될 소멸자 함수 (null이 아닌 경우).

Return: 없음.

Function: 해시 테이블의 모든 원소를 제거한다. 소멸자 함수가 제공되면 각 원소에 대해 호출된다.

**hash\_destroy**

Prototype: void hash\_destroy(struct hash \*h, hash\_action\_func \*destructor)

Parameters:

struct hash \*h – 제거할 해시 테이블.

hash\_action\_func \*destructor – hash를 해제하기 위해 호출될 소멸자 함수 (null이 아닌 경우).

Return: 없음.

Function: 해시 테이블을 파괴하고, 내려받은 소멸자 함수로 나머지 원소를 해제한다.

**hash\_insert**

Prototype: struct hash\_elem \*hash\_insert(struct hash \*h, struct hash\_elem \*new)

Parameters:

struct hash \*h – 원소를 삽입할 해시 테이블.

struct hash\_elem \*new – 삽입할 새 원소.

Return: 새 원소와 같은 기존 원소가 없으면 null, 있으면 해당 원소.

Function: 해시 테이블에 새 원소를 삽입한다. 동등한 원소가 이미 존재하면 삽입하지 않고 기존 원소를 반환한다.

**hash\_replace**

Prototype: struct hash\_elem \*hash\_replace(struct hash \*h, struct hash\_elem \*new)

Parameters:

struct hash \*h – 원소를 교체할 해시 테이블.

struct hash\_elem \*new – 삽입할 새 원소.

Return: 교체될 원소. 교체될 원소가 없으면 null.

Function: 해시 테이블에서 새 원소를 동등한 기존 원소와 교체한다.

**hash\_find**

Prototype: struct hash\_elem \*hash\_find(struct hash \*h, struct hash\_elem \*e)

Parameters:

struct hash \*h – 검색할 해시 테이블.

struct hash\_elem \*e – 찾고자 하는 원소.

Return: 찾은 원소. 없으면 null.

Function: 해시 테이블에서 주어진 원소와 동등한 원소를 찾아 반환한다.

**hash\_delete**

Prototype: struct hash\_elem \*hash\_delete(struct hash \*h, struct hash\_elem \*e)

Parameters:

struct hash \*h – 원소를 제거할 해시 테이블.

struct hash\_elem \*e – 제거할 원소.

Return: 제거한 원소. 없으면 null.

Function: 해시 테이블에서 주어진 원소와 동 등한 원소를 찾아 제거하고 반환한다. 원소가 해시 테이블에 존재하지 않으면 null을 반환한다.

**hash\_apply**

Prototype: void hash\_apply(struct hash \*h, hash\_action\_func \*action)

Parameters:

struct hash \*h – 작업을 적용할 해시 테이블.

hash\_action\_func \*action – 각 원소에 적용할 함수.

Return: 없음.

Function: 해시 테이블의 모든 원소에 주어진 작업을 순서대로 적용한다.

**hash\_first**

Prototype: void hash\_first(struct hash\_iterator \*i, struct hash \*h)

Parameters:

struct hash\_iterator \*i – 초기화할 반복자.

struct hash \*h – 탐색할 해시 테이블.

Return: 없음.

Function: 반복자가 해시 테이블의 첫 번째 원소를 가리키도록 한다.

**hash\_next**

Prototype: struct hash\_elem \*hash\_next(struct hash\_iterator \*i)

Parameters:

struct hash\_iterator \*i – 현재 원소의 반복자.

Return: 다음 원소 반환. 다음 원소가 없으면 null 반환.

Function: 반복자를 다음 원소로 이동시키고, 해당 원소를 반환한다.

**hash\_cur**

Prototype: struct hash\_elem \*hash\_cur(struct hash\_iterator \*i)

Parameters:

struct hash\_iterator \*i – 현재 원소의 반복자.

Return: 현재 반복자가 가리키는 원소를 반환, 없으면 null.

Function: 반복자가 현재 가리키고 있는 원소를 반환한다.

**hash\_size**

Prototype: size\_t hash\_size(struct hash \*h)

Parameters:

struct hash \*h – 원소의 개수를 조회할 해시 테이블.

Return: 해시 테이블에 있는 원소의 개수.

Function: 해시 테이블에 있는 원소의 개수를 반환한다.

**hash\_empty**

Prototype: bool hash\_empty(struct hash \*h)

Parameters:

struct hash \*h – 비어 있는지 확인할 해시 테이블.

Return: 해시 테이블이 비어 있으면 true, 그렇지 않으면 false.

Function: 해시 테이블이 비어 있는지 여부를 true or false로 반환한다.

**hash\_bytes**

Prototype: unsigned hash\_bytes(const void \*buf\_, size\_t size)

Parameters:

const void \*buf\_ – 해시할 데이터.

size\_t size – 데이터의 크기(바이트 단위).

Return: 데이터의 해시 값.

Function: 주어진 바이트 배열의 해시 값을 계산한다.

**hash\_string**

Prototype: unsigned hash\_string(const char \*s\_)

Parameters:

const char \*s\_ – 해시할 문자열.

Return: 문자열의 해시 값.

Function: 주어진 문자열의 해시 값을 계산한다.

**hash\_int**

Prototype: unsigned hash\_int(int i)

Parameters:

int i – 해시할 정수.

Return: 정수의 해시 값.

Function: 주어진 정수의 해시 값을 계산한다.

**hash\_int2**

Prototype: unsigned hash\_int2(int i)

Parameters:

int i – 해시로 변환할 정수.

Return: 정수의 해시 값.

Function: Knuth 곱셈 알고리즘을 사용하여 주어진 정수의 해시 값을 계산한다.

1. **Bitmap**

**bitmap\_create**

Prototype: struct bitmap \*bitmap\_create(size\_t bit\_cnt)

Parameters:

size\_t bit\_cnt – 생성할 비트맵의 비트 개수.

Return: 성공 시 비트맵 포인터, 실패 시 NULL.

Function: 지정된 비트 개수만큼 비트맵을 초기화하고, 모든 비트를 false로 설정하여 초기화한다.

**bitmap\_create\_in\_buf**

Prototype: struct bitmap \*bitmap\_create\_in\_buf(size\_t bit\_cnt, void \*block, size\_t block\_size)

Parameters:

size\_t bit\_cnt – 생성할 비트맵의 비트 개수.

void \*block – 비트맵 저장에 사용될 블록의 주소.

size\_t block\_size – 블록의 크기(바이트 단위).

Return: 성공 시 비트맵 포인터, 실패 시 NULL.

Function: 지정된 저장 공간을 사용하여 지정된 개수의 비트로 비트맵을 생성한다.

**bitmap\_destroy**

Prototype: void bitmap\_destroy(struct bitmap \*b)

Parameters:

struct bitmap \*b – 파괴할 비트맵의 포인터.

Return: None

Function: 파라미터로 받은 비트맵의 저장 공간을 해제하여 파괴한다.

**bitmap\_size**

Prototype: size\_t bitmap\_size(const struct bitmap \*b)

Parameters:

const struct bitmap \*b – size를 알아낼 비트맵 포인터.

Return: 비트맵의 비트 개수.

Function: 비트맵의 크기(비트의 개수)를 반환한다.

**bitmap\_set**

Prototype: void bitmap\_set(struct bitmap \*b, size\_t idx, bool value)

Parameters:

struct bitmap \*b – 설정할 비트맵 포인터.

size\_t idx – 설정할 비트의 인덱스.

bool value – 해당 비트를 설정할 값(true or false)

Return: 없음.

Function: 전달받은 인덱스의 비트를 지정된 값으로 설정한다.

**bitmap\_mark**

Prototype: void bitmap\_mark(struct bitmap \*b, size\_t bit\_idx)

Parameters:

struct bitmap \*b – 설정할 비트맵 포인터.

size\_t idx – 설정할 비트의 인덱스.

Return: 없음.

Function: 전달받은 인덱스의 비트를 true 값으로 설정한다.

**bitmap\_reset**

Prototype: void bitmap\_mark(struct bitmap \*b, size\_t bit\_idx)

Parameters:

struct bitmap \*b – 설정할 비트맵 포인터.

size\_t idx – 설정할 비트의 인덱스.

Return: 없음.

Function: 전달받은 인덱스의 비트를 false 값으로 설정한다.

**bitmap\_flip**

Prototype: void bitmap\_flip(struct bitmap \*b, size\_t bit\_idx)

Parameters:

struct bitmap \*b – 플립할 비트맵의 포인터.

size\_t bit\_idx – 플립할 비트의 인덱스.

Return: 없음.

Function: 전달받은 인덱스의 비트값을 반전시킨다.

**bitmap\_test**

Prototype: bool bitmap\_test(const struct bitmap \*b, size\_t idx)

Parameters:

const struct bitmap \*b – 테스트할 비트맵의 포인터.

size\_t idx – 테스트할 비트의 인덱스.

Return: 전달받은 인덱스의 비트가 1이면 true, 그렇지 않으면 false.

Function: 전달받은 인덱스의 비트가 설정되어 있는지 테스트한다.

**bitmap\_set\_all**

Prototype: void bitmap\_set\_all(struct bitmap \*b, bool value)

Parameters:

struct bitmap \*b – 설정할 비트맵 포인터.

bool value – 설정할 값(true or false).

Return: 없음.

Function: 비트맵의 모든 비트를 value 값으로 설정한다.

**bitmap\_scan**

Prototype: size\_t bitmap\_scan(const struct bitmap \*b, size\_t start, size\_t cnt, bool value)

Parameters:

const struct bitmap \*b – 스캔할 비트맵의 포인터.

size\_t start – 시작 인덱스.

size\_t cnt – 스캔할 연속된 비트의 개수.

bool value – 찾고자 하는 비트의 값.

Return: 조건에 맞는 첫 번째 비트 세트의 시작 인덱스, 그런 세트가 없으면 ERROR 출력

Function: 지정된 값의 연속된 비트 세트를 찾아 그 시작 인덱스를 반환한다.

**bitmap\_expand**

Prototype: struct bitmap\* bitmap\_expand(struct bitmap\* bitmap, size\_t n)

Parameters:

struct bitmap\* bitmap – 확장할 비트맵의 포인터.

size\_t n – 추가할 비트의 개수.

Return: 확장된 비트맵의 포인터.

Function: 비트맵을 확장하여 새로운 비트들을 초기화해준다.

**bitmap\_set\_multiple**

Prototype: void bitmap\_set\_multiple(struct bitmap \*, size\_t start, size\_t cnt, bool)

Parameters:

struct bitmap \* – 수정할 비트맵의 포인터.

size\_t start – 설정 시작 인덱스.

size\_t cnt – 설정할 비트의 개수.

bool – 설정할 값.

Return: 없음.

Function: 지정된 범위의 비트를 특정 값으로 일괄 설정한다.

**bitmap\_count**

Prototype: size\_t bitmap\_count(const struct bitmap \*, size\_t start, size\_t cnt, bool)

Parameters:

const struct bitmap \* – 카운트할 비트맵의 포인터.

size\_t start – 시작 인덱스.

size\_t cnt – 카운트할 비트의 범위.

bool – 카운트할 값.

Return: 지정된 값으로 설정된 비트의 개수.

Function: 지정된 범위 내에서 특정 값으로 설정된 비트의 개수를 반환한다.

**bitmap\_contains**

Prototype: bool bitmap\_contains(const struct bitmap \*, size\_t start, size\_t cnt, bool)

Parameters:

const struct bitmap \* – 검사할 비트맵의 포인터.

size\_t start – 시작 인덱스.

size\_t cnt – 검사할 비트의 범위.

bool – 검사할 값 (true or false).

Return: 지정된 범위 내에 특정 값으로 설정된 비트가 하나라도 있는지 여부.

Function: 지정된 범위 내에서 특정 값으로 설정된 비트가 하나라도 있는지 확인한다.

**bitmap\_scan\_and\_flip**

Prototype: size\_t bitmap\_scan\_and\_flip(struct bitmap \*, size\_t start, size\_t cnt, bool)

Parameters:

struct bitmap \* – 스캔하고 뒤집을 비트맵의 포인터.

size\_t start – 시작 인덱스.

size\_t cnt – 스캔할 연속된 비트의 개수.

bool – 찾고자 하는 비트의 값.

Return: 조건에 맞는 첫 번째 비트 Set의 시작 인덱스, 그런 Set이 없으면 ERROR 반환.

Function: 지정된 값의 비트 Set을 찾아 모두 반대 값으로 flip하고, 그 시작 인덱스를 반환한다.

**bitmap\_file\_size**

Prototype: size\_t bitmap\_file\_size(const struct bitmap \*)

Parameters:

const struct bitmap \* – 파일 크기를 계산할 비트맵의 포인터.

Return: 파일에 저장할 때 필요한 바이트 수.

Function: 비트맵을 파일에 저장할 때 필요한 바이트 수를 계산한다.

**bitmap\_dump**

Prototype: void bitmap\_dump(const struct bitmap \*)

Parameters:

const struct bitmap \* – 출력할 비트맵의 포인터.

Return: 없음.

Function: 비트맵의 내용을 16진수 형태로 변환해 출력한다.