**Pintos Project 0-2: Pintos Data Structure**

|  |  |
| --- | --- |
| 담당 교수 : | 박성용 |
| 학번 : | 20121277 |
| 이름 : | 김주호 |
|  |  |

[Additional Implementation 4](#_Toc84241668)

[parse\_cmd() 4](#_Toc84241669)

[str2int() 4](#_Toc84241670)

[list\_swap() 5](#_Toc84241671)

[list\_shuffle() 5](#_Toc84241672)

[bitmap\_expand() 6](#_Toc84241673)

[hash\_int\_2() 6](#_Toc84241674)

[hash\_hash\_function() 6](#_Toc84241675)

[hash\_less\_function() 7](#_Toc84241676)

[hash\_action\_on\_element() 7](#_Toc84241677)

[hash\_action\_destructor() 8](#_Toc84241678)

[list\_handler() 8](#_Toc84241679)

[hash\_handler() 8](#_Toc84241680)

[bitmap\_handler() 9](#_Toc84241681)

[List 9](#_Toc84241682)

[list\_next() 9](#_Toc84241683)

[list\_end() 10](#_Toc84241684)

[list\_insert() 10](#_Toc84241685)

[list\_splice() 10](#_Toc84241686)

[list\_push\_front() 11](#_Toc84241687)

[list\_push\_back() 11](#_Toc84241688)

[list\_remove() 12](#_Toc84241689)

[list\_pop\_front() 12](#_Toc84241690)

[list\_pop\_back() 12](#_Toc84241691)

[list\_front() 13](#_Toc84241692)

[list\_back() 13](#_Toc84241693)

[list\_reverse() 13](#_Toc84241694)

[list\_sort() 14](#_Toc84241695)

[list\_insert\_ordered() 14](#_Toc84241696)

[list\_unique() 15](#_Toc84241697)

[list\_max() 15](#_Toc84241698)

[list\_min() 16](#_Toc84241699)

[Hash Table 16](#_Toc84241700)

[hash\_init() 16](#_Toc84241701)

[hash\_destroy() 17](#_Toc84241702)

[hash\_insert() 17](#_Toc84241703)

[hash\_find() 18](#_Toc84241704)

[hash\_delete() 18](#_Toc84241705)

[hash\_apply() 18](#_Toc84241706)

[hash\_first() 19](#_Toc84241707)

[hash\_next() 19](#_Toc84241708)

[hash\_int() 19](#_Toc84241709)

[Bitmap 20](#_Toc84241710)

[bitmap\_create() 20](#_Toc84241711)

[bitmap\_destroy() 20](#_Toc84241712)

[bitmap\_set() 21](#_Toc84241713)

[bitmap\_mark() 21](#_Toc84241714)

[bitmap\_reset() 21](#_Toc84241715)

[bitmap\_flip() 22](#_Toc84241716)

[bitmap\_test() 22](#_Toc84241717)

[bitmap\_set\_all() 23](#_Toc84241718)

[bitmap\_set\_multiple() 23](#_Toc84241719)

[bitmap\_count() 23](#_Toc84241720)

[bitmap\_contains() 24](#_Toc84241721)

[bitmap\_any() 24](#_Toc84241722)

[bitmap\_none() 25](#_Toc84241723)

[bitmap\_all() 25](#_Toc84241724)

[bitmap\_scan() 26](#_Toc84241725)

[bitmap\_scan\_and\_flip() 26](#_Toc84241726)

# Additional Implementation

## parse\_cmd()

* Prototype
  + void parse\_cmd(char \*cmd)
* Parameter
  + char \*cmd
* Return
  + none
* Function
  + 사용자로부터 입력받은 command를 의미 단위로 parsing하여 ARGS[]에 저장한다.
  + 이를테면, 사용자로부터 create list list0 를 입력 받았을 때,
    - ARGS[0] = create
    - ARGS[1] = list
    - ARGS[2] = list0
    - 위 처럼 입력 의미 단위로 parsing하여 관리한다.

## str2int()

* Prototype
  + int str2int(const char \*)
* Parameter
  + const char \* str\_num
* Return
  + int형 number
* Function
  + 사용자로부터 입력받은 command는 1차적으로 문자열 형태로 프로그램에 넘어오게 된다.
  + '10'과 같이 string으로 표현된 데이터를 10처럼 int형 자료로 변환한다.

## list\_swap()

* Prototype
  + void list\_swap(struct list\_elem \*elem1, struct list\_elem \*elem2)
* Parameter
  + struct list\_elem \*elem1
  + struct list\_elem \*elem2
* Return
  + none
* Function
  + list element 2개를 인자로 받아, 리스트 내 데이터의 위치를 뒤바꾼다.

## list\_shuffle()

* Prototype
  + void list\_shuffle(struct list \*lst)
* Parameter
  + struct list \*lst (shuffle이 완료된 list)
* Return
  + none
* Function
  + 리스트 내의 원소들을 랜덤하게 뒤섞는 함수이다.

## bitmap\_expand()

* Prototype
  + struct bitmap \*bitmap\_expand(struct bitmap \*bm, int add\_sz)
* Parameter
  + struct bitmap \*bm
  + int add\_sz
* Return
  + struct bitmap\* (expand가 적용된 bitmap)
* Function
  + bitmap이 관리하는 bit size를 add\_sz만큼 더 늘린다.

## hash\_int\_2()

* Prototype
  + unsigned hash\_int\_2(int i)
* Parameter
  + int i
* Return
  + unsigned int (hash 함수를 적용한 값)
* Function
  + 인자 i에 대한 hash 값을 리턴한다.

## hash\_hash\_function()

* Prototype
  + unsigned hash\_hash\_function(const struct hash\_elem \*e, void \*aux);
* Parameter
  + const struct hash\_elem \*e
  + void \*aux
* Return
  + unsigned hash value (hash 함수가 적용된 값)
* Function
  + 인자로 받는 hash element에 대하여 대응하는 hash 값을 리턴한다.

## hash\_less\_function()

* Prototype
  + bool hash\_less\_function(const struct hash\_elem \*a, const struct hash\_elem \*b, void \*aux)
* Parameter
  + const struct hash\_elem \*a
  + const struct hash\_elem \*b
  + void \*aux
* Return
  + bool true or false
* Function
  + hash 원소들에 대한 comparator 기능을 수행한다.

## hash\_action\_on\_element()

* Prototype
  + void hash\_action\_on\_element(struct hash\_elem \*e, void \*aux)
* Parameter
  + struct hash\_elem \*e
  + void \*aux
* Return
  + none
* Function
  + 인자로 입력된 hash element에 대하여 \*aux로 입력된 hash 함수를 적용한다.

## hash\_action\_destructor()

* Prototype
  + void hash\_action\_destructor(struct hash\_elem \*e, void \*aux)
* Parameter
  + struct hash\_elem \*e
  + void \*aux
* Return
  + none
* Function
  + hash element를 제거한다.
  + hash element에 대응하는 hash item이 차지하는 메모리공간을 free 시킨다.

## list\_handler()

* Prototype
  + void list\_handler(char \*, action\_t action)
* Parameter
  + char \*
  + action\_t action
* Return
  + none
* Function
  + 사용자로부터 입력받은 list 관련 command를 핸들링한다.

## hash\_handler()

* Prototype
  + void hash\_handler(char \*, action\_t action)
* Parameter
  + char \*
  + action\_t action
* Return
  + none
* Function
  + 사용자로부터 입력받은 hash 관련 command를 핸들링한다.

## bitmap\_handler()

* Prototype
  + void bitmap\_handler(char \*, action\_t action)
* Parameter
  + char \*
  + action\_t action
* Return
  + none
* Function
  + 사용자로부터 입력받은 bitmap 관련 command를 핸들링한다.

# List

## list\_next()

* Prototype
  + struct list\_elem\* list\_next (struct list\_elem \*elem)
* Parameter
  + struct list\_elem \*elem
* Return
  + struct list\_elem \*elem (next list element)
* Function
  + 인자로 입력받은 list element의 다음 list element를 리턴한다.

## list\_end()

* Prototype
  + struct list\_elem \*list\_end (struct list \*list)
* Parameter
  + struct list \*list
* Return
  + struct list\_elem \*
* Function
  + 입력받은 list의 가장 마지막 list element를 리턴한다.

## list\_insert()

* Prototype
  + void list\_insert (struct list\_elem \*before, struct list\_elem \*elem)
* Parameter
  + struct list\_elem \*before
  + struct list\_elem \*elem
* Return
  + none
* Function
  + 입력받은 before list element의 next 자리에 인자로 주어진 list element를 삽입한다.

## list\_splice()

* Prototype
  + void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last)
* Parameter
  + struct list\_elem \*before
  + struct list\_elem \*first
  + struct list\_elem \*last
* Return
  + none
* Function
  + 인자로 입력받은 before 다음 자리에 list의 first부터 last까지의 element들을 삽입한다.

## list\_push\_front()

* Prototype
  + void list\_push\_front (struct list \*list, struct list\_elem \*elem)
* Parameter
  + struct list \*list
  + struct list\_elem \*elem
* Return
  + none
* Function
  + list의 head에 원소를 삽입한다.

## list\_push\_back()

* Prototype
  + void list\_push\_back (struct list \*list, struct list\_elem \*elem)
* Parameter
  + struct list \*list
  + struct list\_elem \*elem
* Return
  + none
* Function
  + 리스트의 tail에 원소를 삽입한다.

## list\_remove()

* Prototype
  + struct list\_elem \*list\_remove (struct list\_elem \*elem)
* Parameter
  + struct list\_elem \*elem
* Return
  + struct list\_elem
* Function
  + 인자로 입력받은 list element를 제거한다.
  + list element를 제거하면, 바로 그 next 자리에 있던 element가 해당자리로 오게 되는데, 그 element를 리턴한다.

## list\_pop\_front()

* Prototype
  + struct list\_elem \*list\_pop\_front (struct list \*list)
* Parameter
  + struct list \*list
* Return
  + struct list\_elem\*
* Function
  + 리스트의 head 원소를 제거한다

## list\_pop\_back()

* Prototype
  + struct list\_elem \*list\_pop\_back (struct list \*list)
* Parameter
  + struct list \*list
* Return
  + struct list\_elem\*
* Function
  + 리스트의 tail 원소를 제거한다.

## list\_front()

* Prototype
  + struct list\_elem \*list\_front (struct list \*list)
* Parameter
  + struct list \*list
* Return
  + struct list\_elem\*
* Function
  + 리스트 head의 원소를 리턴한다.

## list\_back()

* Prototype
  + struct list\_elem \*list\_back (struct list \*list)
* Parameter
  + struct list \*list
* Return
  + struct list\_elem\*
* Function
  + 리스트 tail의 원소를 리턴한다.

## list\_reverse()

* Prototype
  + void list\_reverse (struct list \*list)
* Parameter
  + struct list \*list
* Return
  + none
* Function
  + 리스트 내의 원소들의 순서를 거꾸로 뒤집는다.

## list\_sort()

* Prototype
  + void list\_sort (struct list \*list, list\_less\_func \*less, void \*aux)
* Parameter
  + struct list \*list
  + list\_less\_func \*less
  + void \*aux
* Return
  + none
* Function
  + 리스트 내의 원소를 정렬한다.
  + 정렬의 기준은 list\_less\_func comparator를 따른다.

## list\_insert\_ordered()

* Prototype
  + void list\_insert\_ordered (struct list \*list, struct list\_elem \*elem, list\_less\_func \*less, void \*aux)
* Parameter
  + struct list \*list
  + struct list\_elem \*elem
  + list\_less\_func \*less
  + void \*aux
* Return
  + none
* Function
  + 함수의 인자로 넘어온 원소를 리스트에 삽입하고, 정렬한다.

## list\_unique()

* Prototype
  + void list\_unique (struct list \*list, struct list \*duplicates, list\_less\_func \*less, void \*aux)
* Parameter
  + struct list \*list
  + struct list \*duplicates
  + list\_less\_func \*less
  + void \*aux
* Return
  + none
* Function
  + 리스트내에 중복된 원소를 제거한다.
  + 인자로 들어온 struct list \*duplicates에 새로운 list를 넘기면, 해당 리스트에 unique 원소들을 삽입하게 된다.

## list\_max()

* Prototype
  + struct list\_elem \*list\_max (struct list \*list, list\_less\_func \*less, void \*aux)
* Parameter
  + struct list \*list
  + list\_less\_func \*less
  + void \*aux
* Return
  + struct list\_elem\* (max값을 가지는 list element)
* Function
  + 리스트 내의 원소 중 최대값의 원소를 리턴한다.

## list\_min()

* Prototype
  + struct list\_elem \*list\_min (struct list \*list, list\_less\_func \*less, void \*aux)
* Parameter
  + struct list \*list
  + list\_less\_func \*less
  + void \*aux
* Return
  + struct list\_elem\* (min값을 가지는 list element)
* Function
  + 리스트 내의 원소 중 최솟값의 원소를 리턴한다.

# Hash Table

## hash\_init()

* Prototype
  + bool hash\_init (struct hash \*h, hash\_hash\_func \*hash, hash\_less\_func \*less, void \*aux)
* Parameter
  + struct hash \*h
  + hash\_hash\_func \*hash
  + hash\_less\_func \*less
  + void \*aux
* Return
  + bool true or false
* Function
  + 파라미터로 들어온 정보들을 적용하여 hash를 초기화한다.

## hash\_destroy()

* Prototype
  + void hash\_destroy (struct hash \*h, hash\_action\_func \*destructor)
* Parameter
  + struct hash \*h
  + hash\_action\_func \*destructor
* Return
  + none
* Function
  + hash를 메모리 공간에서 free시킨다.

## hash\_insert()

* Prototype
  + struct hash\_elem \*hash\_insert (struct hash \*h, struct hash\_elem \*new)
* Parameter
  + struct hash \*h
  + struct hash\_elem \*new
* Return
  + struct hash\_elem\*
* Function
  + hash에 원소를 삽입한다.

## hash\_find()

* Prototype
  + struct hash\_elem \*hash\_find (struct hash \*h, struct hash\_elem \*e)
* Parameter
  + struct hash \*h
  + struct hash\_elem \*e
* Return
  + struct hash\_elem\*
* Function
  + hashmap에서 인자로 주어진 hash element를 찾아 리턴한다.

## hash\_delete()

* Prototype
  + struct hash\_elem \*hash\_delete (struct hash \*h, struct hash\_elem \*e)
* Parameter
  + struct hash \*h
  + struct hash\_elem \*e
* Return
  + struct hash\_elem\*
* Function
  + hashmap에서 인자로 주어진 hash element를 찾아 제거한다.

## hash\_apply()

* Prototype
  + void hash\_apply (struct hash \*h, hash\_action\_func \*action)
* Parameter
  + struct hash \*h
  + hash\_action\_func \*action
* Return
  + none
* Function
  + 주어진 hashmap에 인자로 주어진 hash함수를 적용한다.

## hash\_first()

* Prototype
  + void hash\_first (struct hash\_iterator \*i, struct hash \*h)
* Parameter
  + struct hash\_iterator \*i
  + struct hash \*h
* Return
  + none
* Function
  + hash\_iterator 가 hashmap의 첫번째 원소를 가리킬 수 있도록 작업한다.

## hash\_next()

* Prototype
  + struct hash\_elem \*hash\_next (struct hash\_iterator \*i)
* Parameter
  + struct hash\_iterator \*i
* Return
  + struct hash\_elem\*
* Function
  + hash\_iterator가 현재 가리키고 있는 원소의 다음 원소를 가리키도록 작업한다.

## hash\_int()

* Prototype
  + unsigned hash\_int (int i)
* Parameter
  + int i
* Return
  + unsigned value
* Function
  + 인자 i에 대한 hash 값을 리턴한다.

# Bitmap

## bitmap\_create()

* Prototype
  + struct bitmap \* bitmap\_create (size\_t bit\_cnt)
* Parameter
  + size\_t bit\_cnt
* Return
  + struct bitmap\* (초기화된 bitmap)
* Function
  + 인자로 입력받은 bit\_cnt만큼 bitmap을 초기화한다.

## bitmap\_destroy()

* Prototype
  + void bitmap\_destroy (struct bitmap \*b)
* Parameter
  + struct bitmap \*b
* Return
  + none
* Function
  + 인자로 입력받은 bitmap에 할당된 메모리공간을 free시킨다.

## bitmap\_set()

* Prototype
  + void bitmap\_set (struct bitmap \*b, size\_t idx, bool value)
* Parameter
  + struct bitmap \*b
  + size\_t idx, bool value
* Return
  + none
* Function
  + 인자로 입력받은 위치의 bit를 입력받은 value 값으로 set한다.

## bitmap\_mark()

* Prototype
  + void bitmap\_mark (struct bitmap \*b, size\_t bit\_idx)
* Parameter
  + struct bitmap \*b
  + size\_t bit\_idx
* Return
  + none
* Function
  + 인자로 입력받은 위치의 bit를 입력받은 true 값으로 set한다.

## bitmap\_reset()

* Prototype
  + void bitmap\_reset (struct bitmap \*b, size\_t bit\_idx)
* Parameter
  + struct bitmap \*b
  + size\_t bit\_idx
* Return
  + none
* Function
  + 인자로 입력받은 위치의 bit를 입력받은 false 값으로 reset한다.

## bitmap\_flip()

* Prototype
  + void bitmap\_flip (struct bitmap \*b, size\_t bit\_idx)
* Parameter
  + struct bitmap \*b
  + size\_t bit\_idx
* Return
  + none
* Function
  + 인자로 입력받은 위치의 bit를 flip 시킨다.
  + flip이란 기존의 값이 true이면 false로 reset하고, 기존의 값이 false이면 true로 set하는 동작을 말한다.

## bitmap\_test()

* Prototype
  + bool bitmap\_test (const struct bitmap \*b, size\_t idx)
* Parameter
  + const struct bitmap \*b
  + size\_t idx
* Return
  + bool true or false
* Function
  + bitmap의 특정 index에 설정된 값을 리턴한다.

## bitmap\_set\_all()

* Prototype
  + void bitmap\_set\_all (struct bitmap \*b, bool value)
* Parameter
  + struct bitmap \*b
  + bool value
* Return
  + none
* Function
  + bitmap이 관리하는 모든 범위의 값을 인자로 입력받은 value로 설정한다.

## bitmap\_set\_multiple()

* Prototype
  + void bitmap\_set\_multiple (struct bitmap \*b, size\_t start, size\_t cnt, bool value)
* Parameter
  + struct bitmap \*b
  + size\_t start, size\_t cnt
  + bool value
* Return
  + none
* Function
  + [start] 인덱스부터 cnt개의 원소를 value로 설정한다.

## bitmap\_count()

* Prototype
  + size\_t bitmap\_count (const struct bitmap \*b, size\_t start, size\_t cnt, bool value)
* Parameter
  + const struct bitmap \*b
  + size\_t start
  + size\_t cnt, bool value
* Return
  + size\_t bitmap의 개수
* Function
  + bitmap에서 특정 value(true or false)로 설정된 값이 몇개인지 리턴한다.

## bitmap\_contains()

* Prototype
  + bool bitmap\_contains (const struct bitmap \*b, size\_t start, size\_t cnt, bool value)
* Parameter
  + const struct bitmap \*b
  + size\_t start
  + size\_t cnt
  + bool value
* Return
  + bool true or false
* Function
  + [start] 인덱스부터 cnt개의 원소에서 인자로 입력받은 value의 값을 하고 있는 원소가 있는지 check한다.

## bitmap\_any()

* Prototype
  + bool bitmap\_any (const struct bitmap \*b, size\_t start, size\_t cnt)
* Parameter
  + const struct bitmap \*b
  + size\_t start
  + size\_t cnt
* Return
  + bool true or false
* Function
  + [start] 인덱스부터 cnt개의 원소에서 인자로 입력받은 value의 값을 하고 있는 원소가 하나라도 있다면, true를 리턴한다.

## bitmap\_none()

* Prototype
  + bool bitmap\_none (const struct bitmap \*b, size\_t start, size\_t cnt)
* Parameter
  + const struct bitmap \*b
  + size\_t start
  + size\_t cnt
* Return
  + bool true or false
* Function
  + [start] 인덱스부터 cnt개의 원소에서 bit가 설정되어 있지 않다면(모두 false라면), true를 리턴한다.

## bitmap\_all()

* Prototype
  + bool bitmap\_all (const struct bitmap \*b, size\_t start, size\_t cnt)
* Parameter
  + const struct bitmap \*b
  + size\_t start
  + size\_t cnt
* Return
  + bool true or false
* Function
  + [start] 인덱스부터 cnt개의 원소에서, 모든 원소가 인자로 입력받은 value의 값을 하고 있다면, true를 리턴한다.

## bitmap\_scan()

* Prototype
  + size\_t bitmap\_scan (const struct bitmap \*b, size\_t start, size\_t cnt, bool value)
* Parameter
  + const struct bitmap \*b
  + size\_t start
  + size\_t cnt
  + bool value
* Return
  + size\_t bitmap 원소의 index
* Function
  + [start] 인덱스부터 cnt개의 원소에서 value로 설정된 bit가 있다면, true를 리턴한다.

## bitmap\_scan\_and\_flip()

* Prototype
  + size\_t bitmap\_scan\_and\_flip (struct bitmap \*b, size\_t start, size\_t cnt, bool value)
* Parameter
  + struct bitmap \*b
  + size\_t start
  + size\_t cnt
  + bool value
* Return
  + size\_t bitmap 원소의 index
* Function
  + [start] 인덱스부터 cnt개의 원소를 탐색하다가 value로 설정된 bit가 있다면 해당 bit의 값을 flip시킨다.