**과목명: 시스템프로그래밍**

**1 분반**

**<<Project #1>>**

**서강대학교 [컴퓨터공학과]**

**[20161559]**

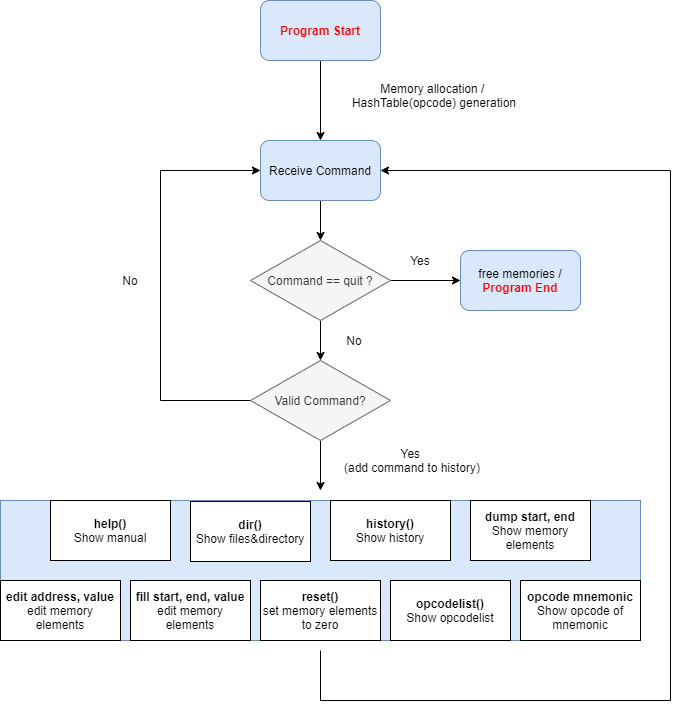
**[강민석]**

목 차

1. **프로그램 개요**
2. **프로그램 설명**
   1. 프로그램 흐름도
   2. 프로그램 파일구성
   3. 사용라이브러리
   4. 사용자 정의 구조체
3. **모듈 정의**
4. **전역 변수 정의**
5. **코드 설명**
6. **프로그램 개요**

SIC/XE 머신을 구현하기 위한 전 단계로, 머신에 가상 메모리 공간을 설계하고 각종 operation에 대한 opcode 변환 및 기본적인 명령어 기능을 갖는 SIC 시뮬레이터 프로그램을 작성한다..

1. **프로그램 설명**
   1. 프로그램 흐름도



프로그램의 흐름은 위와 같다. 20161559.out 파일을 실행하면 1MB의 메모리가 동적으로 할당되고 opcode.txt파일을 바탕으로 opcode 해시테이블을 작성하고 사용자로부터 입력이 들어올 때까지 대기하는 형태이다. 명령어가 입력으로 들어오면 명령어가 유효한지 판단하고, 유효하다면 history에 입력으로 들어온 명령어를 집어넣고, 해당 명령어에 대응되는 함수를 호출하고 다시 입력을 받는 대기상태로 돌아간다. 만약 명령어가 유효하지 않다면 간단한 에러메시지를 출력하고 다시 입력을 받는 상태로 돌아간다. 명령어로 quit(프로그램의 종료)이라는 문자열이 들어오면 프로그램을 실행할 때 동적으로 할당했던 메모리(1MB 가상메모리) 및 해시테이블을 다 free해주고 main함수를 종료함으로써 프로그램이 종료된다.

* 1. 프로그램 파일 구성

구현한 SIC 가상 머신은 20161559.c 파일과 20161559.h 파일로 구성되어있다. c 소스파일에서는 기본적인 프로그램의 흐름 및 메모리와 해시테이블 공간 할당만 해주었고 구체적인 함수 및 에러처리는 헤더파일에 작성하였다.

* 1. 사용 라이브러리

<string.h>

명령어 문자열 복사 및 조작, 문자열 길이 계산을 위해 관련 라이브러리를 사용하였다.

<stdlib.h>

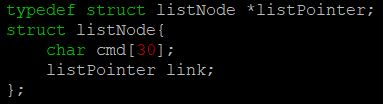
해쉬테이블 및 가상 메모리의 동적할당 및 해제 기능을 위해 사용하였다.

<dirent.h>, <sys/stat.h>

프로그램에서 디렉토리 및 파일 정보를 조회하는 기능을 하는 함수 dir()를 구현하기 위해 사용하였다.

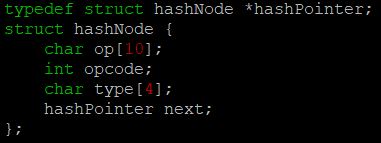
* 1. 사용자 정의 구조체

(1) listPointer



history에 linked list의 형태로 명령어를 삽입하기 위해 정의한 구조체이다.

(2)hashPointer



hashTable의 linked list 부분을 구현하기 위해 정의한 구조체이다.

1. **모듈 정의**

정의한 모듈에 대해 간략적인 설명

**int hasing(char \*)** : 문자열을 매개변수로 받아 0에서 19사이의 해시값을 반환한다.

**char\* lstrim(char \*)** : 문자열의 왼쪽 공백을 없애주어 문자열을 반환한다.

**char\* rtrim(char \*)** : 문자열의 오른쪽 공백을 없애주고 새로운 문자열을 동적으로 할당하여 반환한다.

**char \*trim(char \*)** : ltrim, rtrim함수를 호출하여 문자열의 양쪽 공백을 없애준다.

**void history\_cleaner(void)** : history를 구성하였던 linkedlist 요소들을 전부 free 해줌으로써 heap영역에 있는 메모리를 해제해준다.

**void hashTable\_cleaner(void)** : Hashtable을 구성하였던 linkedlist 요소들을 전부 free 해줌으로써 heap영역에 있는 메모리를 해제해준다.

**void insert\_command(char \*)** : 문자열을 history에 추가하는 기능을 갖는다.

**void help(void)** : 수행가능한 명령어들을 화면에 출력한다.

**void dir(void)** : 현 프로그램이 실행되는 디렉토리를 기준으로 존재하는 파일과 디렉토리 list를 화면에 출력한다.

**void history(void)** : history에 저장되어있는 명령어들을 순서대로 출력한다.

**void dump(int, int)** : 가상 메모리 공간에 저장된 정보를 출력한다.

**void edit(int, int)** : 가상 메모리 공간의 값을 수정하는 함수이다.

**void fill(int, int, int)** : 가상 메모리 공간에 저장되어 있는 값을 범위를 지정해서 수정하는 기능을 한다.

**void reset(void)** : 가상 메모리 공간의 값을 모두 0으로 만든다.

**void opcodelist(void)** : 사용할 수 있는 operation의 목록을 화면에 출력한다.

**int parseHex(char\*, int, int)** : 문자열로 된 16진수를 정수로 바꾸어 반환해준다.

**void commandManager(char\*, char\*)** : 프로그램의 핵심 함수로 명령을 매개변수로 받아 적합한 명령이면 그에 따른 함수를 호출해주고, 그러지 않으면 간단한 에러 메시지를 출력하는 등 프로그램의 전반적인 흐름을 담당하는 함수이다.

1. **전역 변수 정의**

**unsigned char\* memory :** 프로젝트에서 요구하는 1Mbyte의 가상메모리를 구현하기 위해 선언한 전역변수이다.

**listPointer first** : linked list의 첫번째 요소를 가리키기 위한 변수이다.

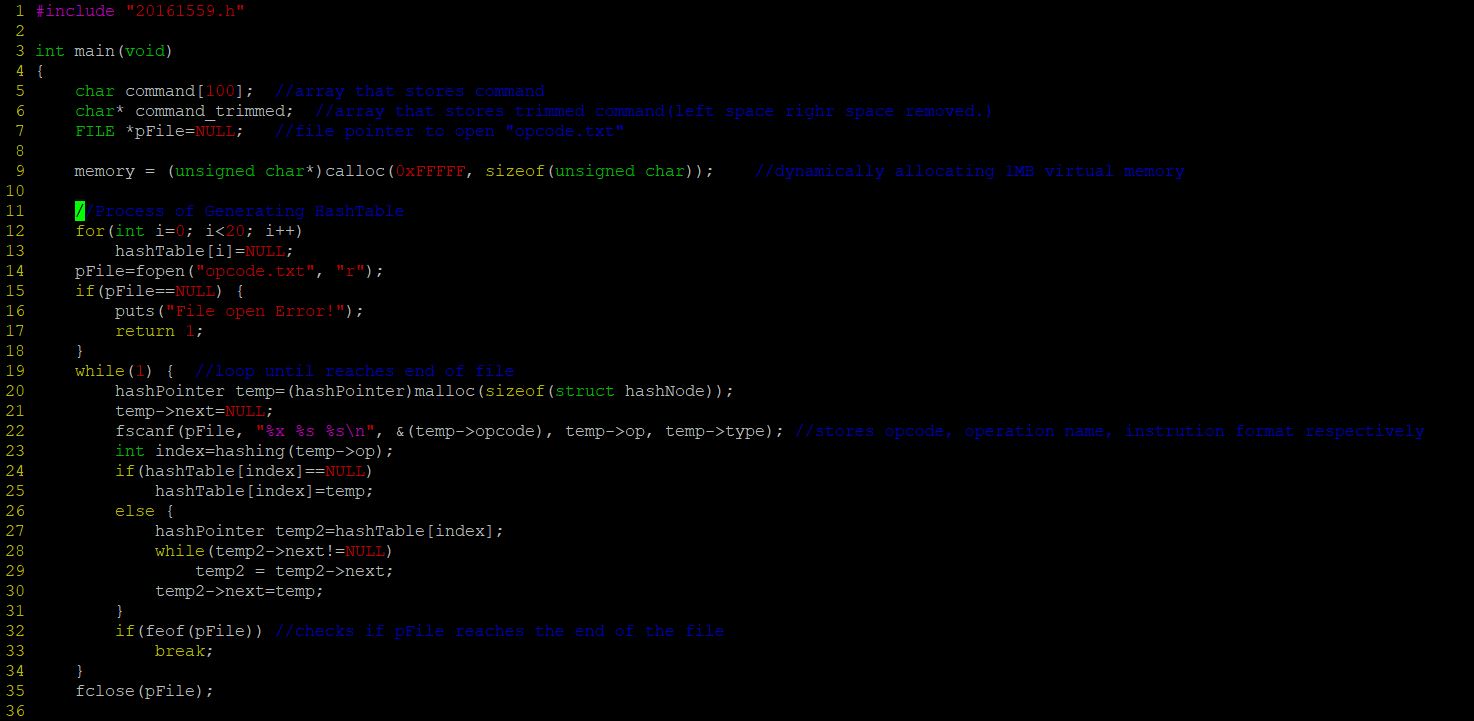
**listPointer trail** : linked list의 마지막 요소를 가리키기 위한 변수이다. (history 삽입에 이용된다.)

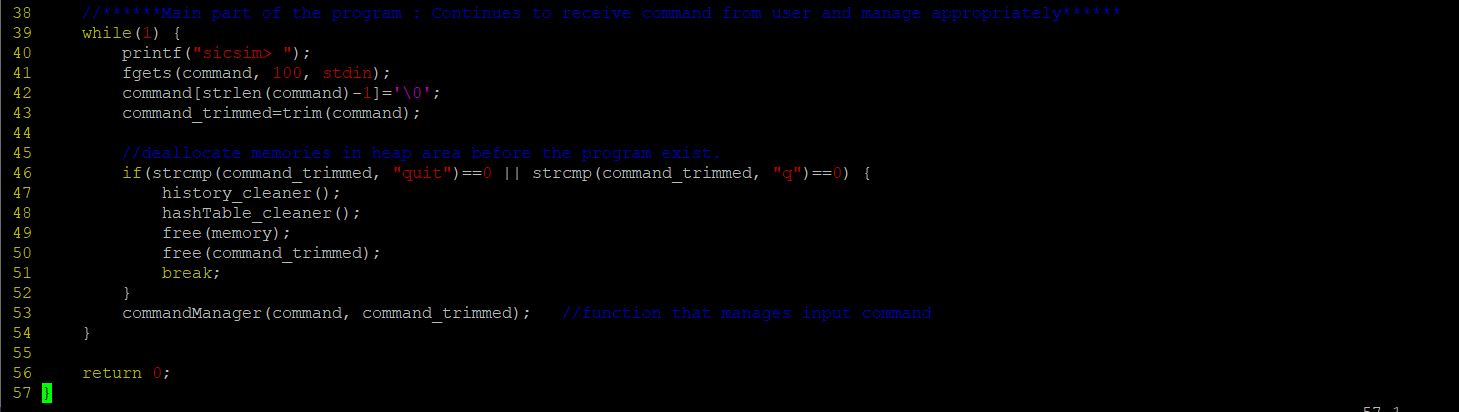
**int start\_address** : dump함수 실행 시 마지막으로 출력되었던 주소+1을 저장하기 위한 변수이다.

**hashPointer hashTable[20]** : 크기가 20인 해시테이블을 위해 선언된 변수이다.

1. **코드 설명**

**- 20161559.c**

****



**line 5 ~ 9** : 기본적인 변수들을 설정한다. **char command[100]**에는 앞으로 입력으로 들어올 명령어가 문자열 형태로 저장이된다. **char\* command\_trimmed**는 trim함수를 command 변수에 대해 실행한 결과값을 저장하는 변수로, 양측의 공백이 제거된 형태의 명령어가 저장이 된다. FILE **\*pFile**은 opcode.txt 파일로부터 데이터를 받아 해시테이블을 구성하기 위한 파일포인터 변수이다. **memory**는 헤더 파일에 선언된 전역변수로 프로그램에서 구현해야 될 1Mbyte의 가상메모리를 동적을 할당해주어 초기세팅을 한다.

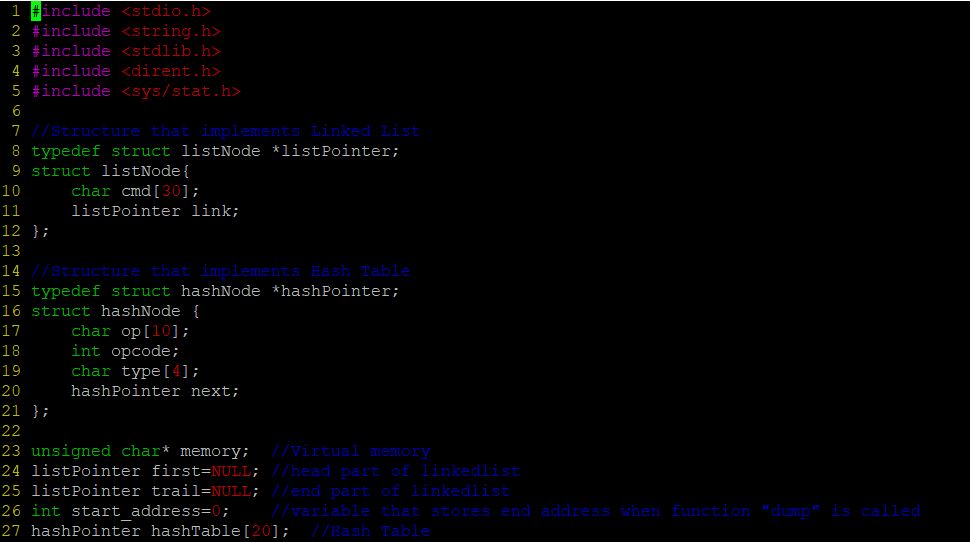
**line 11 ~ 35** : opcode.txt 파일로부터 데이터를 받아 해시테이블을 구성하는 과정이다. 우선 크기가 20인 hashTable이 전역변수로 헤더파일에 선언된 상황에서 fopen함수를 통해 opcode.txt를 읽어들인다. opcode.txt 에는 한 줄 마다

**18 ADD 3/4**

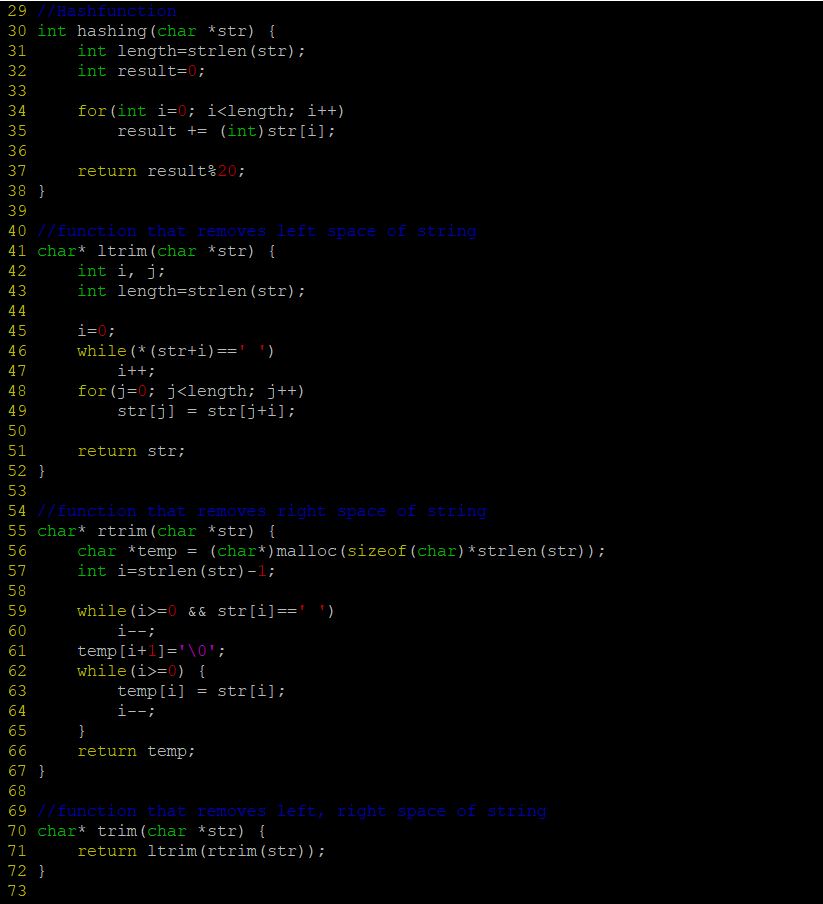
와 같은 형태로 데이터가 있으며 “16진수 / 문자열 / 문자열”의 형태로 fscanf로 받아 hashPointer를 생성하고 hash함수를 통해 나온 index 값에 따라 hashTable에 집어넣는 과정을 반복한다. opcode.txt의 파일 끝에 도달하면 fclose를 통해 파일포인터가 열었던 파일을 닫아준다.

**line 39 ~ 54** : 프로그램의 핵심인 부분이다. while 루프를 통해 항상 화면에는 “sicsim>”형태로 출력을 해주어 sic simulator라는 것을 보여주고 command 변수에 명령어를 fgets 함수로부터 받는다. command\_trimmed에는 입력받은 커맨드의 공백을 처리한 후의 문자열로 command\_trimmed가 “quit” 또는 “q”(프로그램에서 종료를 뜻하는 명령어이다.)와 같으면 동적으로 할당된 메모리를 반환하는 함수들을 호출한 후while 루프를 break하여 main함수의 끝에 도달하고 프로그램이 종료된다. 그렇지 않은 경우 20161559.h에 정의된 commandManager라는 함수를 호출하여 명령어에 따른 적절한 처리를 진행한다.

**- 20161559.h**



**line 1 ~ 27** : 함수 정의 및 구현에 앞서 기본적으로 필요한 라이브러리를 include하고, 앞서 설명한 구조체를 정의하고 필요한 전역변수들을 선언한다.



**line 30 ~ 38** : hashTable에 필요한 hash 함수이다. 매개변수로부터 문자열을 받아, 각각의 character ASCII 값을 더한 후 20으로 나눈 나머지 값을 반환한다. 해시테이블의 index가 0에서 19 사이의 값을 가지므로 hashTable 크기의 범위를 만족한다.

**line 41 ~ 52** : 매개변수로 들어온 문자열의 왼쪽 공백을 모두 없애는 기능을 하는 함수이다. while 루프를 통해 공백이 처음으로 나타나지 않는 인덱스를 왼쪽에서부터 진행하여 찾은 후, 그 index로부터 length 까지 str[j] = str[j + i]를 for 루프를 통해 왼쪽으로 끌어당기는 역할을 한다. for 루프가 끝나면 해당 문자열을 반환한다.

**line 55 ~ 67** : 매개변수로 들어온 문자열의 오른쪽 공백을 모두 없애는 기능을 하는 함수이다. 여기서 이후에 command\_trimmed가 적절한 명령어로 판단되어 history에 삽입될 때, command\_trimmed가 아니라 command, 즉 가공되기 이전의 문자열을 history에 삽입하기 위해 rtrim함수에서는 동적으로 새로운 문자열을 할당해서 return 한다. 구체적인 과정은 매개변수로 들어온 문자열 str의 오른쪽에서부터 시작해서 처음으로 공백이 나타나지 않는 index를 while루프를 통해 발견한 후, temp[i] = str[i]연산을 통해 temp에 str의 원소를 하나하나 복사하고 temp문자열을 반환해준다.

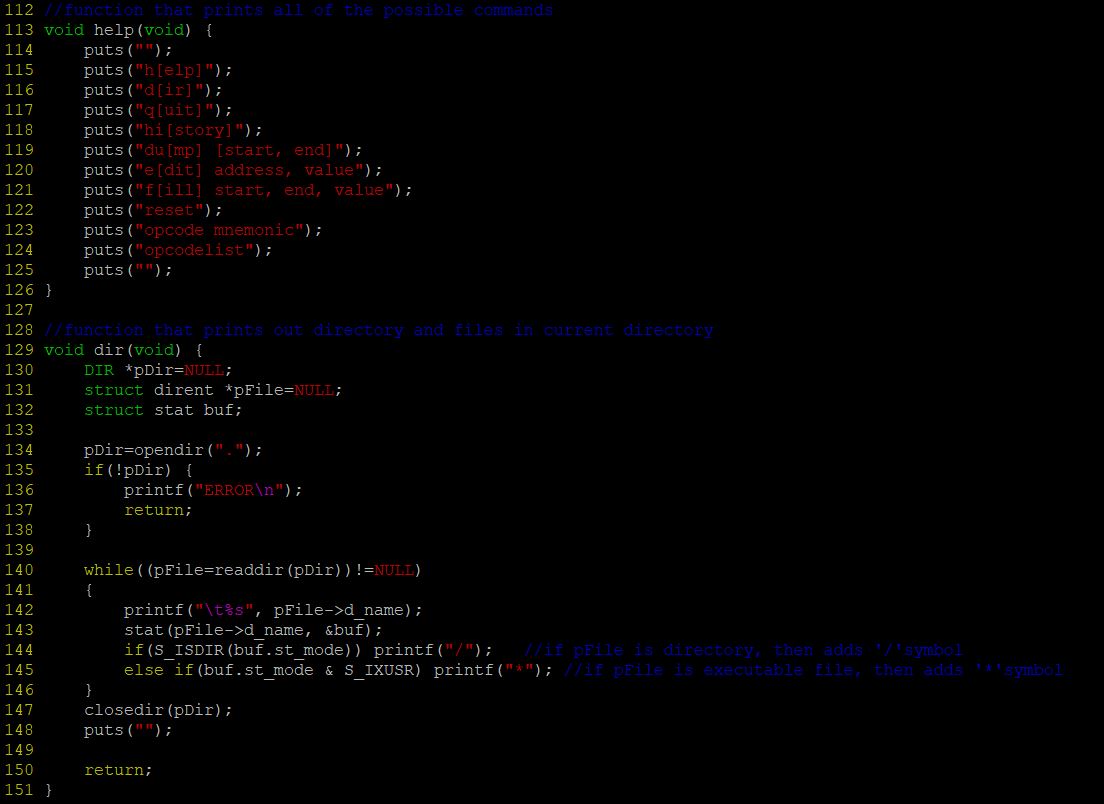
**line 70 ~ 72** : 매개변수 문자열의 양쪽 공백을 모두 없애주는 기능을 갖는 함수로 먼저 rtrim을 호출한 뒤, ltrim을 호출하는 방식으로 구성되어있다.



**line 75 ~ 83** : history에 있는 linked list를 모두 free 해주는 함수이다. 프로그램이 종료되기 직전에 호출되며, 전역변수로 선언된 first값을 temp변수에 저장한 후 앞에서부터 하나하나 free 해주는 while 루프로 구성되어있다.

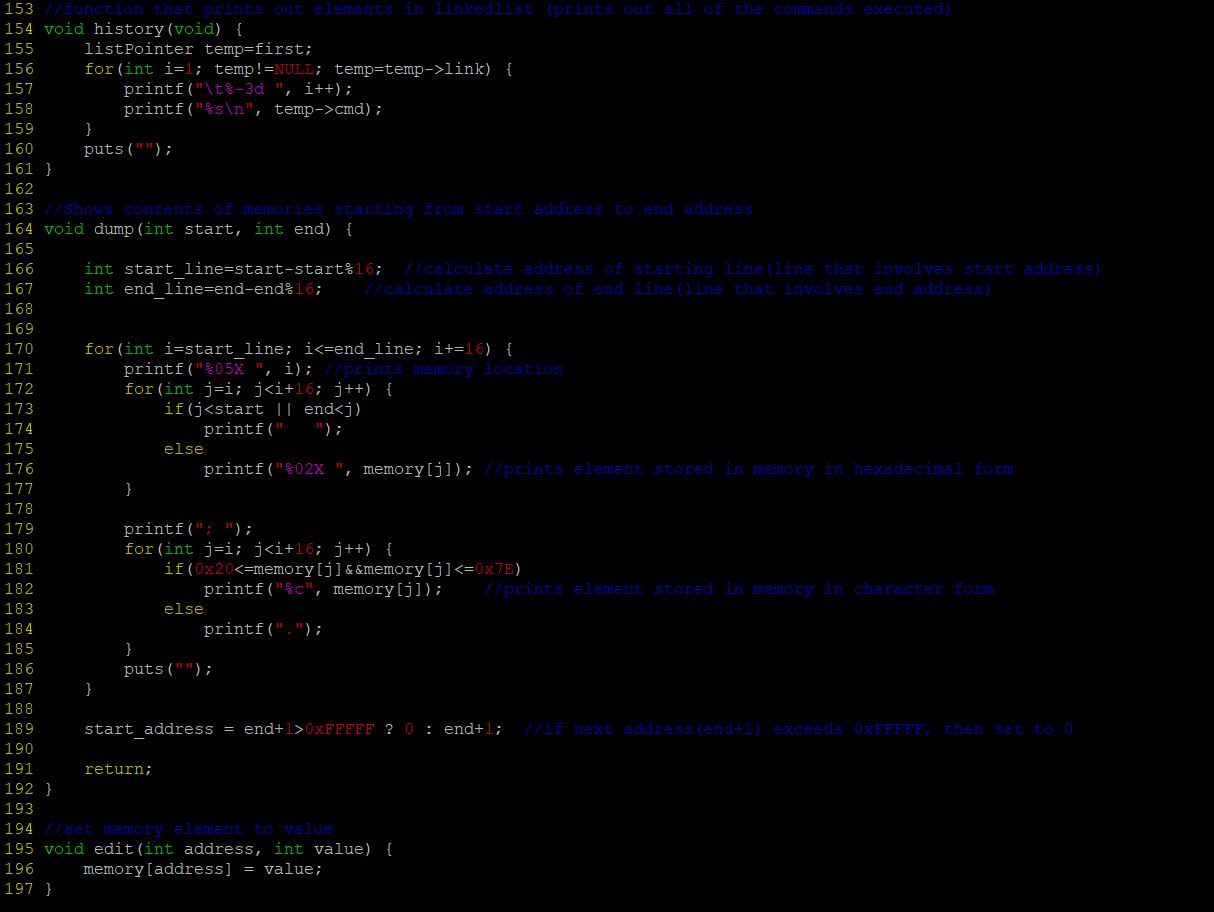
**line 86 ~ 96** : hashTable에 있는 linked list를 모두 free 해주는 함수이다. 프로그램이 종료되기 직전에 호출되며, hashTable의 크기인 20만큼 각 index마다 연결된 hashPoitner들을 앞에서부터 하나씩 free 해주는 형태의 반복문으로 구성되어있다.

**line 99 ~ 110** : 함수가 호출되기 직전 입력으로 들어왔던 명령어가 유효한 명령어로 판별나서 history에 저장하는 과정이다. history가 linked list로 만들어졌으므로 새로운 listPointer를 동적으로 할당해주어 명령어를 strncpy를 통해 복사하고 history를 구성하는 linekd list의 마지막 부분에 삽입하는 기능을 한다.



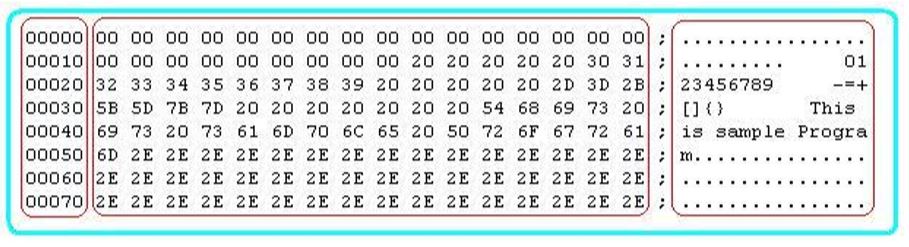
**line 113 ~ 126** : 함수가 호출되면 프로그램에서 실행 가능한 명령어들을 모두 출력해준다.

**line 129 ~ 151** : 함수가 호출되면 프로그램이 실행되는 디렉토리의 모든 파일과 디렉토리를 출력하는 기능을 한다. 우선 pDir로부터 현재 디렉토리를 의미하는 “.”을 opendir한 후, while 루프를 통해 모든 파일 및 디렉토리를 탐색한다. while 루프 안에서는 우선 파일의 이름을 출력하고 해당 파일이 디렉토리이면 추가적으로 “/(슬래시)”를 출력한다. 그렇지 않고 만약 실행가능한 파일이면(line 145) “\*”을 옆에 추가로 출력해준다. 모든 기능을 다 수행했으면 closedir를 통해 열었던 디렉토리를 닫아준다.



**line 154 ~ 161** : history에 저장되어 있는 명령어들을 모두 출력하는 함수이다. 함수가 호출되면 for 루프를 통해 listPointer temp 변수가 linked list의 앞부분부터 차례대로 순회하면서 내용을 출력해주는 기능을 한다.

**line 164 ~ 192** : 프로그램에서 구현한 가상 메모리 공간에 담겨있는 내용을 출력하는 함수이다. 매개변수로 start와 end를 가지며, start부터 end 주소까지 담겨있는 내용을 출력한다. 변수 start\_line과 end\_line은 출력 형태가 아래와 같기 때문에



start와 end를 포함하는 메모리 line을 구하기 위해 사용된다. (위 첨부사진 가장 왼쪽부분) 가장 바깥쪽 for 루프는 출력할 line수를 결정하며 안쪽의 첫 번째 for 루프는 메모리에 담겨있는 내용을 16진수 두 자리 형태로 출력해준다. (위 첨부사진 중간 부분) 안쪽의 두 번째 for 루프는 메모리에 담겨 있는 내용을 아스키코드 문자로 출력해준다. (위 첨부사진 가장 우측부분) 만약에 메모리에 담겨있는 내용이 16진수 20과 7E사이의 범위를 만족하지 못하면 “.”으로 출력한다. 모든 루프를 실행한 후, dump 명령어는 이후에 다시 실행되면 이전에 실행했을 때의 마지막 주소(end)이후부터 출력하므로 해당 내용을 start\_address에 저장한다. 삼항연산자를 이용하여 만약 다음 주소값이 범위를 넘어가게 되면(0xFFFFF초과의 값) 0으로 다시 설정하여 처음부터 출력될 수 있도록 지정해준다.

**line 195 ~ 197** ; 가상 메모리에 저장되어있는 내용을 수정하는 함수로 매개변수로 주어진 주소값에 있는 내용을 매개변수 value로 수정한다.



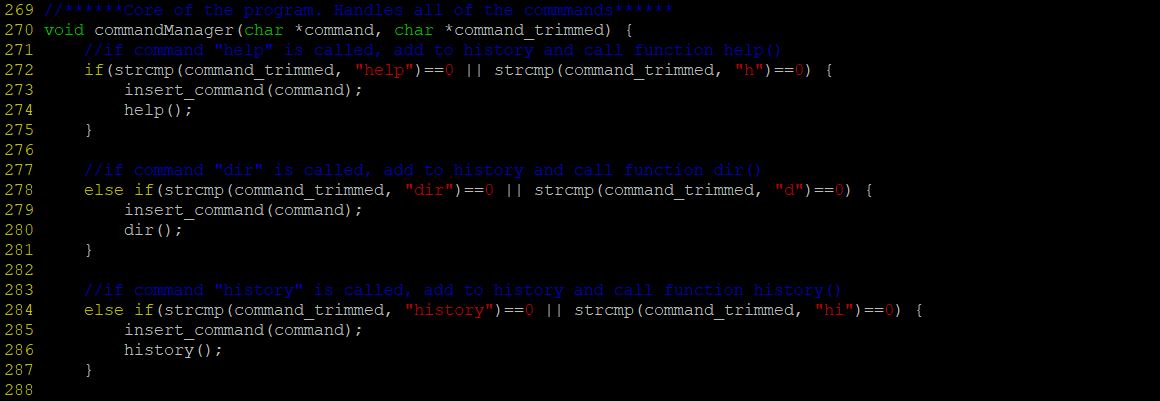
**line 200 ~ 203** : 매개변수 start 부터 매개변수 end까지의 주소값에 저장되어있는 메모리의 내용을 value로 수정하는 함수이다.

**line 206 ~ 209** : 가상메모리에 있는 모든 값을 0으로 초기화시키는 함수이다.

**line 212 ~ 229** : 사용가능한 operation과 그 operation의 opcode를 출력하는 함수이다. hashTable에 있는 값을 반복문을 통해 차례대로 순회하면서 출력한다.



**line 232 ~ 267** : 문자열 형태로 입력받은 16진수를 정수로 반환하는 형태이다. 여기서 str은 command\_trimmed 그 자체이므로, start index부터 end index 까지의 값만 고려하는 방식이다. line 237과 line 239의 while 루프는 trim함수와 유사하게 공백을 모두 제거하는 기능을 갖는다. 그 후 start 부터 end 까지 아스키 값을 고려하여 result에 값을 차례대로 더하여 최종적으로 반환하는 형태이다. 만약에 해당부분에 16진수가 아닌 문자가 들어가 있으면 (ex) 1HF) 그 즉시 -1을 return함으로써 parseHex를 호출한 부분에서 -1을 반환 받는다면 16진수 값이 없음을 파악할 수 있다.

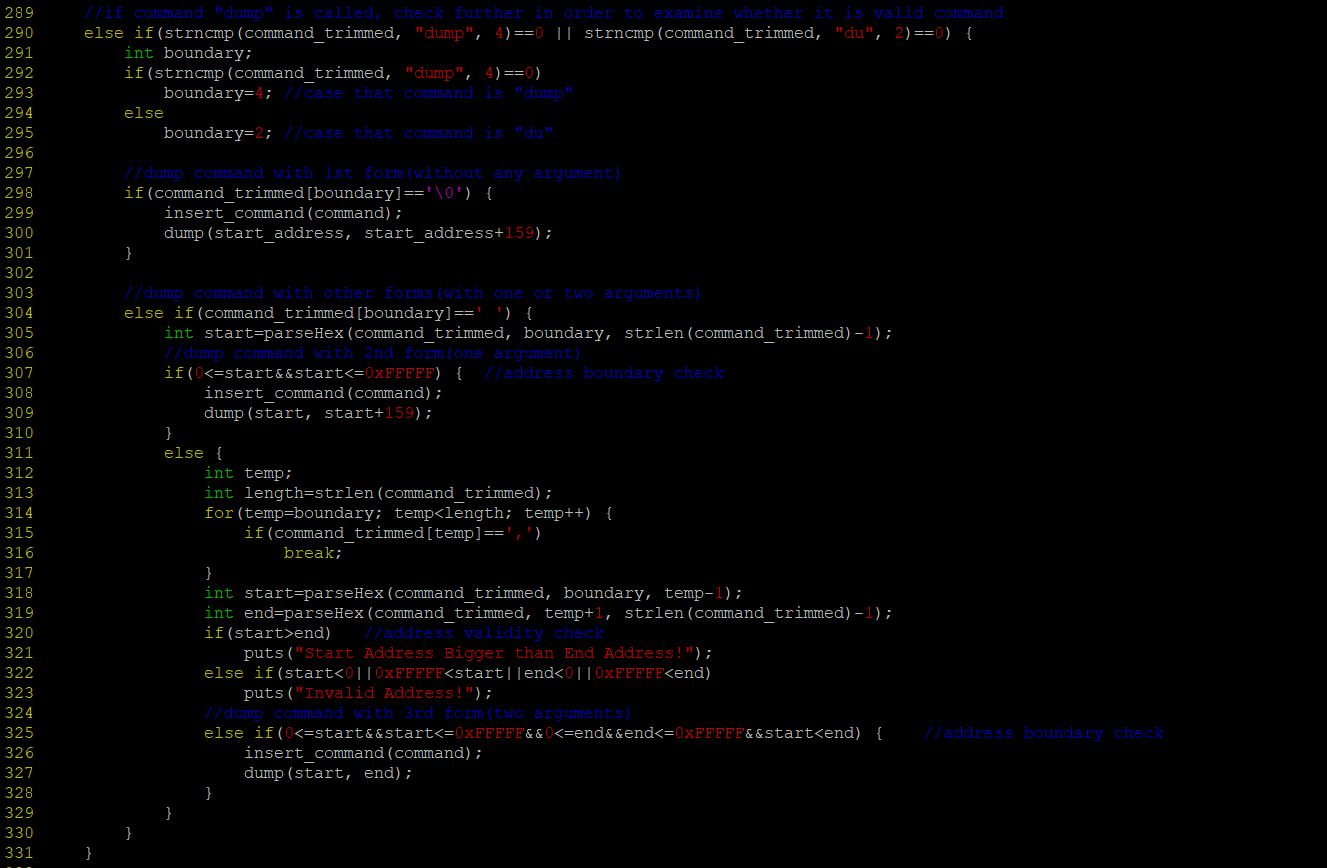


**line 270** : 프로그램의 가장 중요한 함수인 commandManager이다. 두 개의 매개변수 command와 command\_trimmed를 받아, command\_trimmed가 유효한 명령어이면 command를 history에 추가하고 명령어에 대응되는 함수를 호출하는 기능을 수행한다.

**line 272 ~ 275** : help 명령어가 입력으로 들어오면 history에 추가하고 help함수를 호출한다.

**line 278 ~ 281** : dir 명령어가 입력으로 들어오면 history에 추가하고 dir함수를 호출한다.

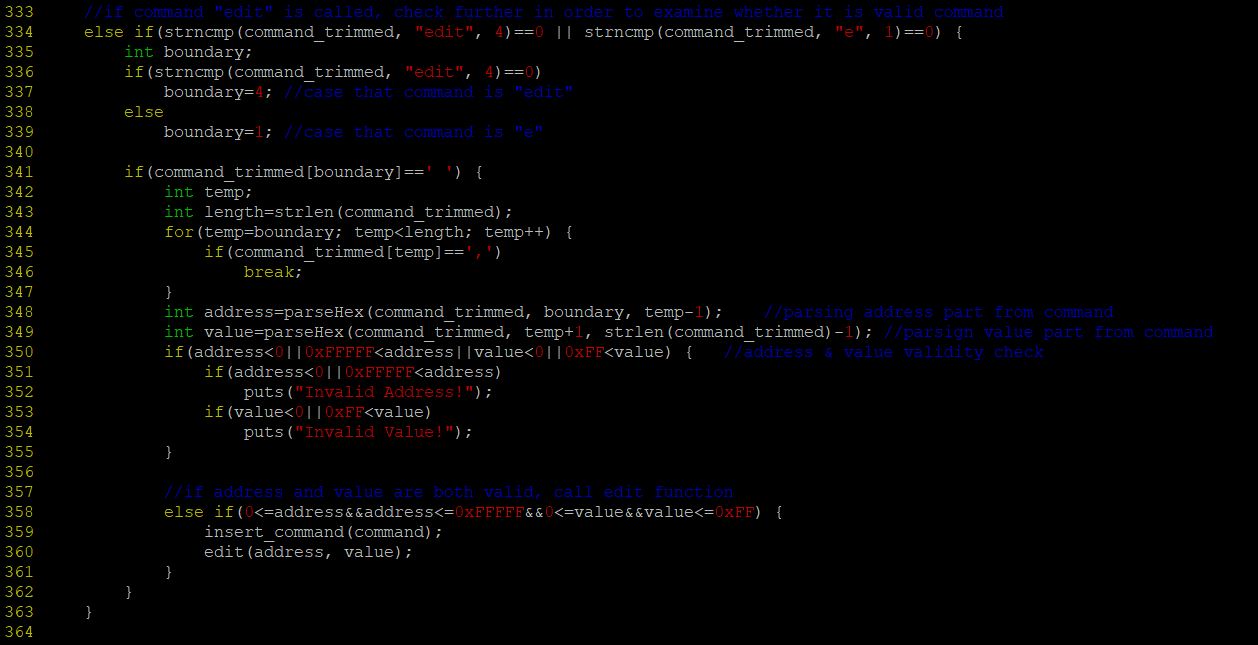
**line 283 ~ 287** : history 명령어가 입력으로 들어오면 history에 추가하고 history함수를 호출한다.



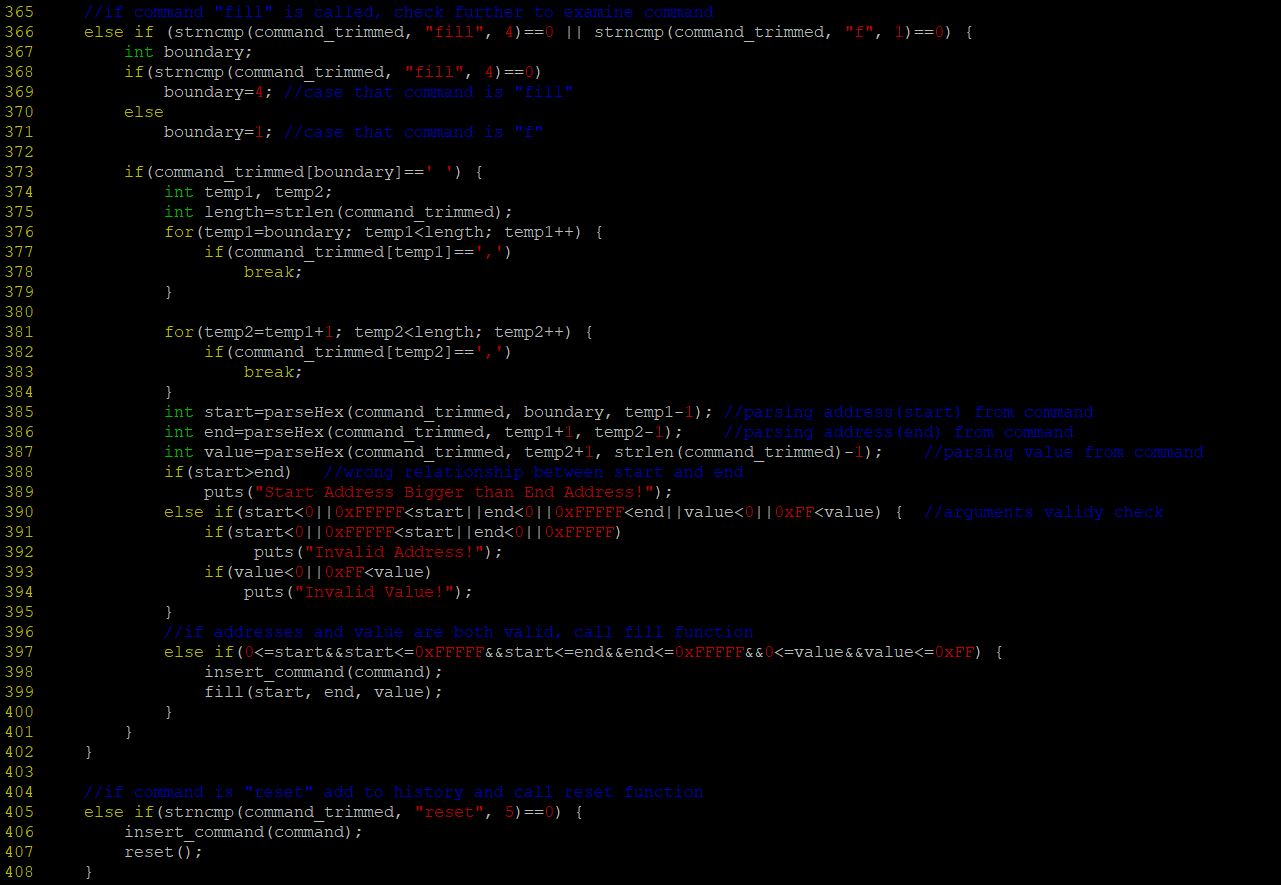
**line 290 ~ 296** : dump 명령어는 dump 또는 du 형태로 입력을 받는다. boundary 변수는 해당 명령어가 dump인지 du인지 구분하는 용도로 사용된다.

**line 298 ~ 301** : dump 명령어가 첫 번째 형태, 즉 추가 매개변수 없이 “dump”와 같이 단독으로 들어오는 형태이다. 처음부터 160개의 주소를 출력해주면 되므로 dump(start\_address 부터 start\_address+159)를 호출해준다.

**line 304 ~ 341** : 우선 dump가 두 번째 형태(매개변수 하나만 동반하는 dump 10과 같은 형태)인지 검사한다. start 변수에 parseHex함수를 호출하여 16진수 값으로 표현될 수 있음이 확인되면 dump를 호출하고 그렇지 않은 경우 dump가 세 번째 형태(매개변수 두 개를 동반하는 dump 10, 20과 같은 형태)인지 검사한다. 두 매개변수를 구분짓는 ‘,(반점)’의 index를 찾아서 그 index를 기준으로 parseHex함수를 호출하여 반점 앞 뒤가 모두 16진수 형태로 표현되는지 확인한다. 그 후 가상메모리 주소 값의 범위를 벗어나는지 확인한 후, 벗어나지 않는 다면 dump(start, end)를 호출한다. 주소 값의 범위를 벗어나는 경우 간단한 에러메시지를 출력한다.

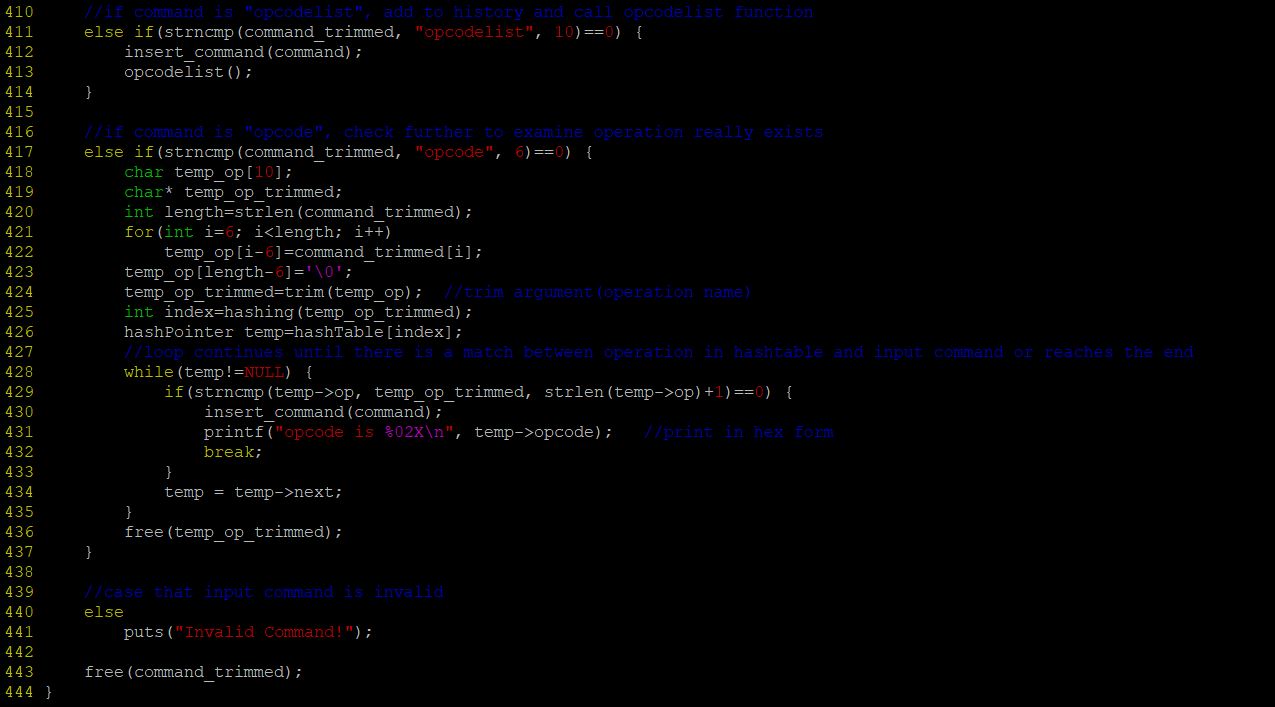


**line 334 ~ 339** : dump와 유사하게 edit은 “edit”혹은 “e” 두 가지 형태로 입력을 받을 수 있다. boundary는 그 둘을 구분 짓는 용도이다. edit 명령어는 항상 두 가지 매개변수를 동반하므로 바로 “,(반점)”의 index를 찾은 후, 그 index를 기준으로 parseHex함수를 호출하여 16진수로 표현가능한 형태인지 확인한다. 16진수로 표현이 가능하면 앞의 매개변수는 주소값이므로 0xFFFFF이하의 값을 갖는지, 뒤의 매개변수는 value이므로 0xFF의 값을 갖는지 확인한 후, 모든 조건을 만족하면 edit(addresss, value)형태로 함수를 호출해준다.



**line 336 ~ 402** : fill 명령어는 “fill”과 “f” 두 가지 형태로 입력을 받는다. boundary 변수는 그 둘을 구분 짓기 위한 용도이다. fill은 3개의 매개변수를 가지므로 “,(반점)”의 index를 찾은 후, 그 index를 기준으로 3개의 매개변수가 16진수 형태로 나타내어지는지 검사한다. 16진수이면 앞의 두 매개변수는 주소값이므로 0부터 0xFFFFF사이의 값을 갖는지, 그리고 세 번째 매개변수는 value이므로 0부터 0xFF사이의 값을 갖는지 확인한 후, fill(start, end, value)형태로 함수를 호출한다.

**line 405 ~ 408** : reset함수는 별도의 매개변수가 없으므로 들어온 명령어가 “reset” 문자열과 동일하다면 reset 함수를 호출한다.



**line 411 ~ 414** : 명령어가 “opcodelist”문자열과 동일하다면 opcodelist()함수를 호출한다.

**line 417 ~ 443** : 명령어의 앞부분이 “opcode”와 동일하다면 유효한 명령어인지 추가적으로 검사한다. temp\_op\_trimmed는 opcode 뒤에 나오는 문자열을 양쪽의 공백을 제거한 형태로 hashTable을 while루프로 순회하면서 동일한 operation이 있는지 확인한다. 만약에 동일한 operation이 있다면 해당 operation과 opcode를 16진수 형태로 출력해주는 기능을 구현하였다.

**line 440 ~ 443** : 위에서 기술된 어떠한 명령어 문자열과 대응되지 않는 경우 적합하지 않은 명령어로 간주하여 에러메시지인 “Invalid Command”를 출력하고, command\_trimmed는 trim 함수로부터 새롭게 동적 할당된 문자열이므로 free 해주도록 한다.