

HOC 프로그래밍

최종 보고서

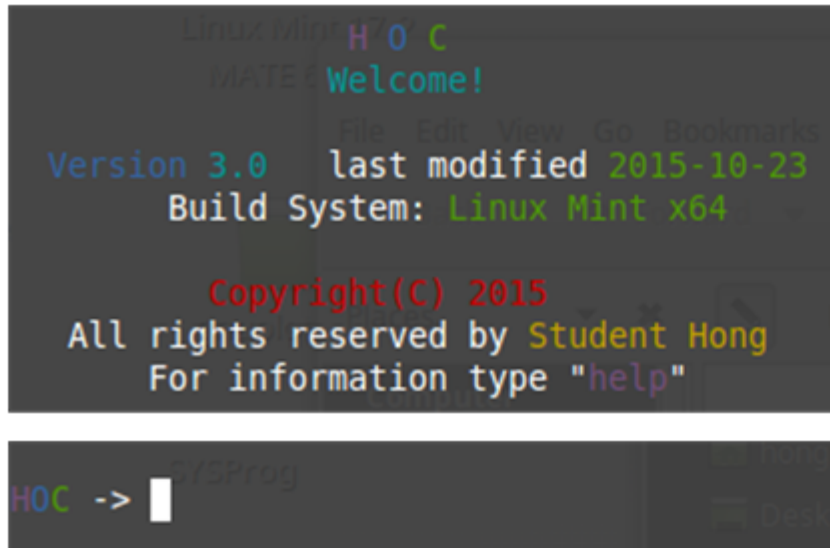
학과	컴퓨터공학과
학번	201211704
이름	김기홍

목 차

1. HOC 프로그램의 개요	3
- HOC 프로그램 시작화면	
- 프로그램에 대한 전반적인 설명	
2. HOC 프로그램의 다양한 기능	4
- 기본적인 사칙연산	
- modulus 연산	
- 단항 연산	
- 전위, 후위 증감연산	5
- 약수의 개수를 구하는 연산	
- 내장함수, 상수 초기화	6
- Symbol table의 효율적인 탐색	
- 난수를 구하는 Rand() 함수	7
- 임시변수 ans	
- 프롬프트 출력	
- 프로그램 설명을 위한 help 명령	8
- 화면을 지우기 위한 clear 명령	
- 함수의 그래프를 그리기 위한 graph 명령	9
3. 주요 기능들의 설명	11
- 약수의 개수를 구하는 연산	
- 임시변수 ans	
- 프롬프트 출력	
- 프로그램 설명을 위한 help 명령	
- 화면을 지우기 위한 clear 명령	
- 함수의 그래프를 그리기 위한 graph 명령	12

1. HOC 프로그램의 개요

- HOC 프로그램 시작화면



```
Linux Mint H2O C
MATE Welcome!
File Edit View Go Bookmarks
Version 3.0 last modified 2015-10-23
Build System: Linux Mint x64

Copyright(C) 2015
All rights reserved by Student Hong
For information type "help"

HOC -> |
```

: HOC 프로그램의 버전과 최종 수정일, 개발 환경, 저작권등의 정보가 담겨있는 메시지를 출력한다.

- 프로그램에 대한 전반적인 설명

: HOC 프로그램은 간단한 계산기로 시작하여 조금씩 기능이 추가되어 왔으며, 현재 3.0 Version의 경우 다양한 연산들과 여러 함수들, 그리고 함수의 그래프도 그릴 수 있다. 추후에는 프로그램 내에서 C언어를 이용해 함수를 작성하는 기능이 추가될 예정이다. 업데이트는 계속 이루어질 것이며 기존의 기능들 또한 더 강력해질 것이다.

2. HOC 프로그램의 다양한 기능

- 기본적인 사칙연산

: HOC 프로그램은 출발이 계산기였던 만큼 기본적인 사칙연산 기능을 가지고 있다.
사칙연산에 대한 우선순위에도 충실한 것을 알 수 있다.

HOC -> 1+3 ans = 4	HOC -> 1+2*3 ans = 7
HOC -> 7-3 ans = 4	HOC -> 4/2*6 ans = 12
HOC -> 2*16 ans = 32	HOC -> (5+7)*3 ans = 36
HOC -> 27/3 ans = 9	HOC -> <input type="text"/>

- modulus 연산

: HOC 프로그램에서 mod 연산은 실수도 피연산자로 가능하다. 구현은 피연산자인 두 수가 모두 정수인지를 검사하고 정수라면 modulus 계산을 한다. 하지만 정수가 아니라면 두 피연산자에 각각 10을 곱하여 다시 검사하고, 정수가 될 때까지 10을 곱한 횟수를 count한다. 거기에 modulus 된 값에 count만큼 10으로 다시 나누어 주면 실수 피연산자도 modulus 연산이 가능하게 된다.

HOC -> 3 % 2 ans = 1	HOC -> 3.7 % 2 ans = 1.7
HOC -> 17 % 5 ans = 2	HOC -> 6 % 1.9 ans = 0.3
HOC -> 387 % 13 ans = 10	HOC -> 2.7 % 1.5 ans = 1.2
HOC -> 210 % 10 ans = 0	HOC -> 8.3 % 3.1 ans = 2.1

- 단항 연산

: 어떤 값에 부호를 정해주는 연산으로서 계산기라면 필수로 가지고 있어야하는 기능 중 하나 이다.

HOC -> a=3	HOC -> -a-b
HOC -> b=5	ans = -8
HOC -> -a	ans = 2
ans = -3	
HOC -> -b	
ans = -5	

- 전위,후위 증감연산

: HOC 프로그램에서는 증감연산자의 결과 출력이 독특하다. 변수명만을 입력했을 때는 해당 변수의 값만 출력되지만, 증감연산을 했을 경우는 **해당 변수의 값과 계산 결과 값도 같이 출력**해준다.

구현은 증감연산의 토큰을 따로 만들고 표준입력의 첫 버퍼가 '+'나 '-'일 때 다음 버퍼를 검사해보고 또 '+' 나 '-'일 경우 토큰을 리턴해주고 아니라면 ungetc를 이용하여 표준입력 버퍼로 한 글자를 다시 돌려주고 빠져나오는 방식으로 구현하였다.

HOC -> a	a = 5		
HOC -> ++a	a = 6	ans = 6	
HOC -> a++	a = 6	ans = 6	
HOC -> a	a = 7		

HOC -> a	a = 8		
HOC -> b	b = 7		
HOC -> a++ - --b	a = 8	b = 6	ans = 2
HOC -> a++ + b--	a = 9	b = 6	ans = 15

- 약수의 개수를 구하는 연산

: HOC 프로그램에는 어떤 수의 약수의 개수를 계산 해주는 연산이 가능하다. 이 기능은 약수의 개수를 구하는만큼 그 개수가 2개인지 아닌지에 따라 **소수를 판별**해낼 수도 있는 연산이다.

```

HOC -> _4
      ans = 3
HOC -> _8
      ans = 4
HOC -> _120
      ans = 16
HOC -> _5785161
      ans = 2
  
```

- 내장함수, 상수 초기화

: 계산에 필요한 함수와 상수들이 미리 선언되어 있다. 그리고 미리 선언된 상수는 값 변경이 불가능하도록 구현하였는데, 방법은 미리 선언된 상수에 따로 CONST 토큰을 만들어서 CONST에 assign 동작을 하면 값 대입이 아닌 에러 메시지를 출력하도록 하였다.

```
HOC -> PI
ans = 3.1415927
HOC -> PHI
ans = 1.618034
HOC -> DEG
ans = 57.29578
HOC -> GAMMA
ans = 0.57721566
```

```
HOC -> log10(100)
ans = 2
HOC -> sqrt(100)
ans = 10
HOC -> log(E)
ans = 1
HOC -> exp(log(10))
ans = 10
```

```
HOC -> PI=3.14
./hoc3: Can't assign a value near line 13
HOC -> GAMMA = PI*E
./hoc3: Can't assign a value near line 14
HOC -> PHI = DEG
./hoc3: Can't assign a value near line 15
```

- Symbol table의 효율적인 탐색

: 아래와 같이 Symbol table의 리스트를 출력해보면 시작 알파벳이 같은 변수끼리 리스트를 이루고 있는 것을 확인할 수 있다. 단순히 첫 글자만으로 나누는 것만으로도 기존의 탐색 성능보다 24배 향상시킬 수 있다. 구현 방법은 크기가 24인 *Symbol형 배열을 만들어 각 인덱스와 알파벳을 사상시켜주었다. 그리고 각 배열의 인덱스는 해당 알파벳으로 시작하는 변수들의 리스트를 가리키는 head로 만들어 구현하였다.

```
HOC -> a
a ans abs atan
a = 0
HOC -> ab
ab a ans abs atan
ab = 0
HOC -> abc
abc ab a ans abs atan
abc = 0
```

```
HOC -> b
b
b = 0
HOC -> c
c cos
c = 0
HOC -> d
d DEG
d = 0
```

- 난수를 구하는 Rand() 함수

: 난수를 출력해내는 함수이다. 오른쪽 사진은 함수를 10000번 호출했을 때 0~100까지 숫자의 분포도를 나타낸 것이다. 아주 균일한 것을 볼 수 있다. 구현은 **의사난수 알고리즘**을 이용하였다.

<pre>HOC -> Rand() ans = 0.191438 HOC -> Rand() ans = 0.987885 HOC -> Rand() ans = 0.001347 HOC -> Rand() ans = 0.108826 HOC -> Rand() ans = 0.949362</pre>	<pre>96 106 99 98 103 112 104 93 107 93 127 122 92 99 104 101 99 95 97 114 95 93 89 103 80 92 102 95 104 98 92 88 120 120 104 98 97 108 90 105 109 97 98 97 102 118 79 76 101 107 98 113 106 115 100 102 84 103 115 86 100 95 85 92 99 108 99 103 109 95 107 71 101 101 93 108 119 109 97 105 100 97 100 88 103 104 88 108 108 116 89 103 108 81 104 91 89 94 92 101</pre>
--	--

```
x ^= x>>11;
x ^= x<<7 & 0x9D2C5680;
x ^= x<<15 & 0xEFC60000;
x ^= x>>18;
return 0.000001*((x&0x7FFFFFFF)%1000000); <-----의사난수 생성 코드
```

- 임시변수 ans

: 임시변수 ans는 대입연산이 아닌 입력 즉, 1+2나 a+b*c와 같은 입력의 결과 값을 저장한다. 일종의 **최근 결과값 버퍼**라고 볼 수 있다.

<pre>HOC -> 1+3*7-8+3 ans = 17 HOC -> ans ans = 17 HOC -> ans*3 ans = 51 HOC -> ans ans = 51</pre>	<pre>HOC -> 1+3 ans = 4 HOC -> ans+3 ans = 7 HOC -> ans*7 ans = 49 HOC -> ans/8 ans = 6.125</pre>
--	---

- 프롬프트 출력

: 식을 입력할 수 있는 상태를 알리기 위한 **프롬프트 기능**이 있다.

<pre>HOC -> HOC -> HOC -> main.c math1.c HOC -> HOC -> HOC -></pre>	<pre>HOC -> (hoc2 ./hoc3: syntax error near line 20 HOC -> 1+3 ans = 4 HOC -> a=2;b=2; HOC -></pre>
---	--

- 프로그램 설명을 위한 **help** 명령

: help를 입력한 모습이다.

```
HOC -> help

S/SProg

HELP FOR INFORMATION

There are many functions beyond a calculator in HOC
HOC can get the answer about a simple operation

Also HOC can display the graph of functions on 2 ways
If you want draw a graph, Please type 'graph'
And then you can see the description to draw a graph

You must be confused with lots of operations
I'm sure that because HOC can make you get into the swing
Just then, type 'clear'
Your screen will be clear!

Thank you for using my program

JUST FUN

HOC ->
```

- 화면을 지우기 위한 **clear** 명령

: 왼쪽은 clear 명령을 입력하기 전이고 오른쪽은 입력한 후이다.

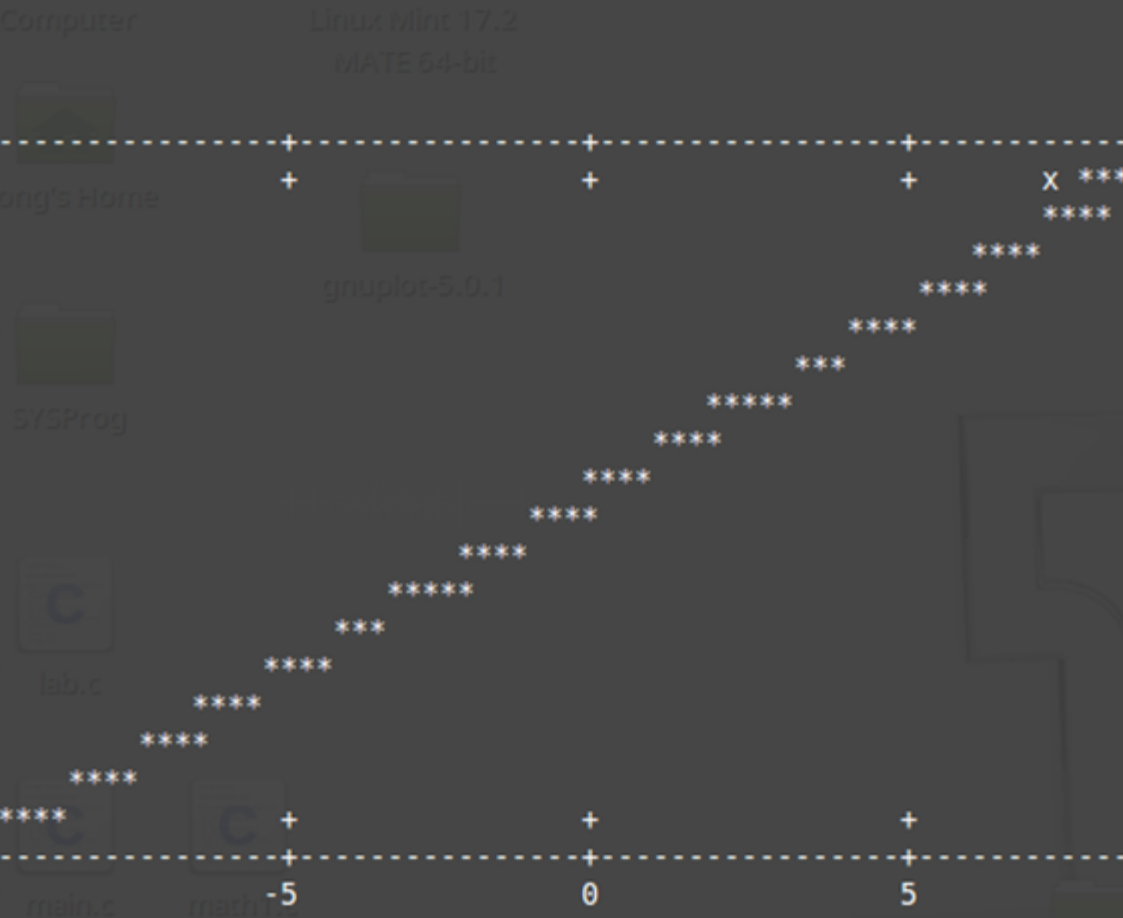
```
HOC -> a=2
HOC -> b=2
HOC -> a+b      ans = 4
HOC -> ans+3    ans = 7
HOC -> ans*3+a  ans = 23
HOC -> clear
```

```
HOC ->
```


: graph 명령이다. graph를 사용하기 위해선 일단 `sudo apt-get install gnuplot` 명령어로 **gnuplot**을 설치해야한다. 아주 가벼운 프로그램이라 부담이 없다. 그리고 또 조금 더 나은 그래픽을 보기 위해서는 **imagemagick**를 설치하면 HOC 프로그램 내에서 입력한 함수의 그래프를 보는 것이 가능해진다.

Let's draw the **graph**!

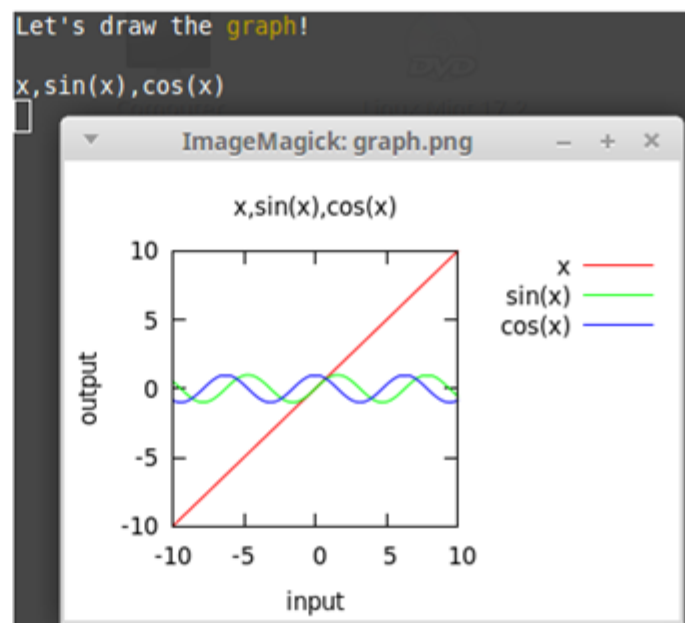
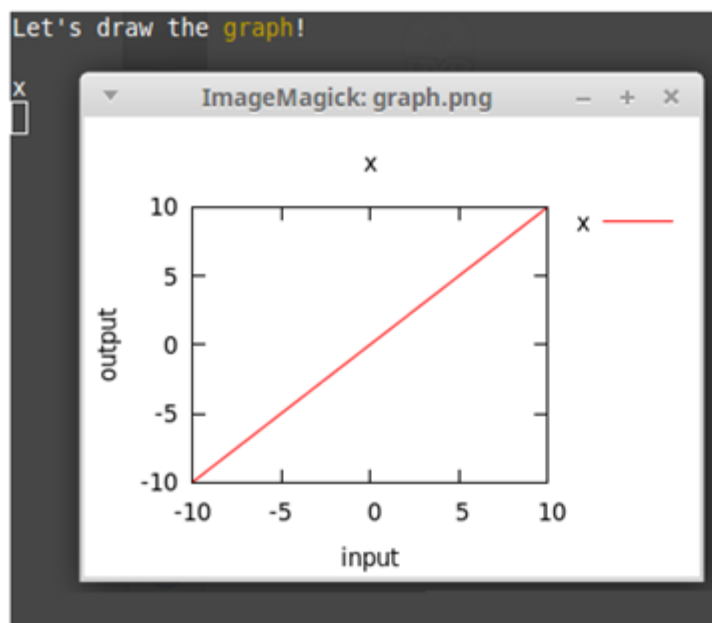
x



gnuplot-5.0.1

you can display the **better graph** by installing imagemagick

HOC ->



Let's draw the graph!

```
x
sh: 1: gnuplot: not found
Please install gnuplot
HOC ->
```

```
select : 3
Please select one of above list
HOC ->
```

```
select : 2
find: `/usr/bin/display': No such file or directory
HOC ->
```

3. 주요 기능들의 설명

- 약수의 개수를 구하는 연산

: 연산은 '.'을 이용하여 구현은 반복문을 사용하여 쉽게 구할 수 있었다. count 변수를 두어 **최초에 2로 초기화** 해주고(1과 자기 자신) 입력값의 제곱근만큼 반복하며 입력 값이 **제곱수라면 +1**을 해주고 그냥 **나누어지는 경우에는 +2**를 해주었다. 이 기능은 단순히 약수의 개수만 보기 보다는 앞에서도 말 했듯이 **소수 판별이 가능하다는 데** 강점이 있다.

- 임시변수 **ans**

: ans라는 변수를 미리 만들어둔 다음 ans 변수가 저장된 노드의 주소값을 가리키는 전역 포인터 변수를 선언하고, **모든 입력 동작이 있을 때 결과 값을 ans 변수에 저장**되도록 하고 출력해주었다. 문제점은 ans가 아닌 일반 변수를 입력하였을 때는 ans가 아니라 그 변수의 이름을 출력해주어야했는데, 이 문제는 변수만을 입력했을 경우에 조건문으로 **변수의 이름이 ans인지 아닌지를 구분**하여 처리해낼 수 있었다.

- **프롬프트** 출력

: yylex()함수 처음에 프롬프트를 출력 해주면 쉽게 구현할 수 있을 줄 알았지만, 문제점이 생각보다 많이 생겼다. 제일 처음 문제점은 yylex() 처음에 그냥 출력했기 때문에 입력 가능 상태일 때만 프롬프트가 생기는 것이 아니라 결과를 출력할 때도 프롬프트가 출력되었다. 또 번갈아 가며 출력하자니 예를 들어 엔터만 쳤을 경우(결과가 없을 경우)에는 또 문제가 발생하였다. 그래서 전역변수 sw를 두고 **조건을 걸어 프롬프트를 출력**하고 **조건이 만족하지 않도록 sw를 토글** 시킨다. 그리고 **프롬프트가 나와야 하는 명령 전에 sw값을 토글** 시키는 방법으로 구현하였다.

- 프로그램 설명을 위한 **help** 명령

: help 명령은 help.txt 파일을 같이 두어 yylex() 내에서 **첫 문자가 알파벳일 때, 전체 표준입력 버퍼를 검사**해서 help와 일치한다면, help.txt의 내용을 터미널에 띄우는 방식으로 구현하였다.

- 화면을 지우기 위한 **clear** 명령

: **help 명령과 마찬가지로** 표준입력 버퍼와 clear이 일치한다면 시스템 함수를 이용해 화면을 지울 수 있었다.

- 함수의 그래프를 그리기 위한 **graph** 명령

: HOC 프로그램의 **최고 기능**이라 할 수 있다. 이 기능은 다른 프로그램(gnuplot, imagemagick)을 **프로그램 내에서 연동**할 수 있다. 표준입력 버퍼의 내용이 graph와 일치한다면 gnuplot에서 그래프를 그리도록 **명령하는 파일의 틀을 생성**하고, 그릴 그래프에 대한 **입력값을 필요한 오프셋에 끼워 넣음**으로서 파일을 완성한다. 그 파일을 통해 gnuplot을 실행시키면 그래프가 화면에 나타나는 식으로 구현하였다.