

2013 Fall Semester SMP

Introduction

Why you're in here?

▶ 지극히 개인적인 입장

- ▶ 포항공대까지 온 애들이 왜 도대체 이걸 못하냐
- ▶ 교수님 수업을 듣고 왜 도대체 이해를 못하냐
- ▶ 랩 과제를 왜 도대체 못 짜냐

▶ 절대로 어려운게 아닙니다

- ▶ 미적분 하는 초등학생보다 코딩하는 초등학생이 많음
- ▶ 여기는 한국에서 공부 잘한다는 1% 모아놓은 곳
- ▶ 자부심을 가져도 되는데 동시에 책임감이 있어야 함...



Why you're in here?

▶ ~~지극히 개인적인 입장~~

- ▶ ~~포항공대까지 온 애들이 왜 도대체 이걸 못하냐~~
- ▶ ~~교수님 수업을 듣고 왜 도대체 이해를 못하냐~~
- ▶ ~~랩 과제를 왜 도대체 못 짜냐~~

▶ 절대로 어려운게 아닙니다

- ▶ 미적분 하는 초등학생보다 코딩하는 초등학생이 많음
- ▶ 여기는 한국에서 공부 잘한다는 1% 모아놓은 곳
- ▶ 자부심을 가져도 되는데 동시에 책임감이 있어야 함...



Why you're in here?

▶ SMP의 장점

- ▶ Programming을 처음 접하면서 이걸 왜 하는지 알 수 있다
- ▶ 나의 필요에 따라서 Programming을 이용할 수 있게 된다
- ▶ 좋은 사람에게 잘 배워서 A+을 받을 수 있다

▶ But, 너무 기대지 말 것

- ▶ SMP는 수업을 제대로 이해하기 위한 Mentoring Program
- ▶ 전 절대로 Assignment 짜주지 않습니다 (나쁜멘토...)
- ▶ 하지만 잘 따라오면 A0 이상은 보장합니다
 - 작년 멘티 4명 중 3명이 A대!! (나머지 1명은 계속 쳐 자서 -_-)



Course Object

- ▶ Programming (Coding) 잘하기
 - ▶ Programming을 잘 하려면 도대체 뭐 어떻게 해야하나?
 - ▶ 언어만 알면 되는거 아닌가?



Course Object

- ▶ **Programming (Coding) 잘하기**
 - ▶ Programming을 잘 하려면 도대체 뭐 어떻게 해야하나?
 - ▶ 언어만 알면 되는거 아닌가?
- ▶ **Problem Solving Design 잘하기**
 - ▶ CS101의 궁극적인 목표
 - ▶ Computer Engineer 가 되기 위한 Paradigm 정립
 - ▶ 먼소리?



Course Object

- ▶ 자연수 p 가 있다. 이 p 가 소수인지 판별하라.



Course Object

- ▶ 자연수 p 가 있다. 이 p 가 소수인지 판별하라.
 - ▶ p 의 약수가 1과 자기자신밖에 없다면 소수, 그렇지 않으면 소수가 아니다.



Course Object

- ▶ 자연수 p 가 있다. 이 p 가 소수인지 판별하라.
 - ▶ p 의 약수가 1과 자기자신밖에 없다면 소수, 그렇지 않으면 소수가 아니다.
- ▶ **So simple!**



Course Object

- ▶ 자연수 p 가 있다. 이 p 가 소수인지 판별하라.
 - ▶ p 의 약수가 1과 자기자신밖에 없다면 소수, 그렇지 않으면 소수가 아니다.
- ▶ 다음의 숫자들 중에서 소수는 무엇인가?

3 7 11 12387482173 9238581273647



Course Object

- ▶ 자연수 p 가 있다. 이 p 가 소수인지 판별하라.
 - ▶ p 의 약수가 1과 자기자신밖에 없다면 소수, 그렇지 않으면 소수가 아니다.
- ▶ 다음의 숫자들 중에서 소수는 무엇인가?

3 7 11 12387482



Course Object

- ▶ **Programming (Coding) 잘하기**
 - ▶ Programming을 잘 하려면 도대체 뭐 어떻게 해야하나?
 - ▶ 언어만 알면 되는거 아닌가?
- ▶ **Problem Solving Design 잘하기**
 - ▶ CS101의 궁극적인 목표
 - ▶ Computer Engineer 가 되기 위한 Paradigm 정립
 - ▶ 어떠한 문제가 주어졌을 때 이를 **해결하기 위한 과정**을 설계
 - ▶ 순서도 및 알고리즘 만들기!



Course Object

- ▶ 자연수 p 가 있다. 이 p 가 소수인지 판별하라.
 - ▶ p 의 약수가 1과 자기자신밖에 없다면 소수, 그렇지 않으면 소수가 아니다.
- ▶ 다음의 숫자들 중에서 소수는 무엇인가?

3 7 11 12387482173 9238581273647



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 p 와 같아지면 Stop



Course Object

- ▶ Programming은 단지 수단일 뿐
 - ▶ 여러분의 목적이 Programming이 되어서는 절대로 안됨
 - ▶ Programming이 절대 어려운 것도 아님.
 - 초딩들도 다 함
 - 비트컴퓨터학원 1개월만 다니면 할 수 있음
- ▶ 따라서 이번 SMP의 목표가 Programming은 절대 아님
 - ▶ C 문법은 아마 한 달 안에 다 배울 수 있을 듯
 - ▶ 코드 예제를 보고 이게 무엇을 하는 프로그램인지 인지하고
 - ▶ 어떠한 문제가 주어졌을 때 이를 해결하는 효율적인 디자인을 찾아내는 연습을 할 것



Course Object

- ▶ Programming은 단지 수단일 뿐
 - ▶ 여러분의 목적이 Programming이 되어서는 절대로 안됨
 - ▶ Programming이 절대 어려운 것도 아님.
 - 초딩들도 다 함
 - 비트컴퓨터학원 1개월만 다니면 할 수 있음
- ▶ 따라서 이번 SMP의 목표가 Programming은 절대 아님
 - ▶ C 문법은 아마 한 달 안에 다 배울 수 있을 듯
 - ▶ 코드 예제를 보고 이게 무엇을 하는 프로그램인지 인지하고
 - ▶ 어떠한 문제가 주어졌을 때 이를 해결하는 **효율적인** 디자인을 찾아내는 연습을 할 것



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2로 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 $p-1$ 와 같아지면 Stop

x 가 갖는 값은 2부터 $p-1$ 까지 총 $p-2$ 개의 숫자를 가짐



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 $p-1$ 와 같아지면 Stop

x 가 갖는 값은 2부터 $p-1$ 까지 총 $p-2$ 개의 숫자를 가짐

따라서 $p-2$ 번의 판단으로 p 가 소수인지 아닌지 결정할 수 있음



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 **root p**와 같아지면
Stop



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 **root p**와 같아지면 Stop

앞의 알고리즘과 비슷하지만 x 는 2부터 \sqrt{p} 까지의 값을 가짐



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 **root p**와 같아지면 Stop

앞의 알고리즘과 비슷하지만 x 는 2부터 \sqrt{p} 까지의 값을 가짐

따라서 $\sqrt{p} - 1$ 번만 비교해보면 됨



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 **root p**와 같아지면 Stop

P 가 16이면 앞의 알고리즘은 14번, 우리의 알고리즘은 4번 비교



Course Object

- ▶ 문제를 해결하기 위한 과정 설계
 - ▶ x 를 2라 하자
 - ▶ 만약 p 가 x 로 나누어 떨어진다면 p 는 소수가 아니다.
 - ▶ 나누어 떨어지지 않는다면 x 를 1 증가시킨다.
 - ▶ 위의 과정을 계속해서 반복하다가 x 가 **root p**와 같아지면 Stop

P 가 16이면 앞의 알고리즘은 14번, 우리의 알고리즘은 4번 비교

100000000이면 앞의 알고리즘은 99999998번 비교하고
우리의 알고리즘은 10000번 비교함

Course Object

- ▶ Programming은 단지 수단일 뿐
 - ▶ 여러분의 목적이 Programming이 되어서는 절대로 안됨
 - ▶ Programming이 절대 어려운 것도 아님.
 - 초딩들도 다 함
 - 비트컴퓨터학원 1개월만 다니면 할 수 있음
- ▶ 따라서 이번 SMP의 목표가 Programming은 절대 아님
 - ▶ C 문법은 아마 한 달 안에 다 배울 수 있을 듯
 - ▶ 코드 예제를 보고 이게 무엇을 하는 프로그램인지 인지하고
 - ▶ 어떠한 문제가 주어졌을 때 이를 해결하는 **효율적인** 디자인을 찾아내는 연습을 할 것



Course Object

- ▶ 하지만 CS101 은 여기까지 오지 않습니다
 - ▶ 내가 무엇을 해결해야 하는지 “문제”를 인지하고
 - ▶ 이 문제를 어떻게 해결해야 하는지 파악하며
 - ▶ 이를 Code로 옮길 수 있다!

- ▶ 하지만 저의 SMP 는 여기까지만 가지 않습니다(?)
 - ▶ 어디까지 가는지는 배워보면 알 수 있습니다 ☺
 - ▶ 사실 그래야 A0까진 받을 수 있어요...



Programming

- ▶ 컴퓨터에게 무언가를 시키는 일
 - ▶ 컴퓨터에게 한국어 해도 못 알아들음
 - ▶ 컴퓨터가 아는 언어로 이야기 해 주는게 Programming
- ▶ 컴퓨터의 언어는 여러 가지가 있음
 - ▶ Assembly Language
 - ▶ Fortran
 - ▶ C/C++
 - ▶ Java
 - ▶ Python
 - ▶ ...



Programming

- ▶ **Computer-friend language**
 - ▶ 컴퓨터가 이해하기 쉬운 언어
 - ▶ 11011100 10110110 01101101 ...
 - ▶ Assembly Language
- ▶ **Human-friend language**
 - ▶ 우리가 이해하기 쉬운 언어
 - ▶ C/C++



Programming

- ▶ Computer-friend language

- ▶ 컴퓨터가 이해하기 쉬운 언어
- ▶ 11011100 10110110 01101101 ...
- ▶ Assembly Language

- ▶ Human-friend language

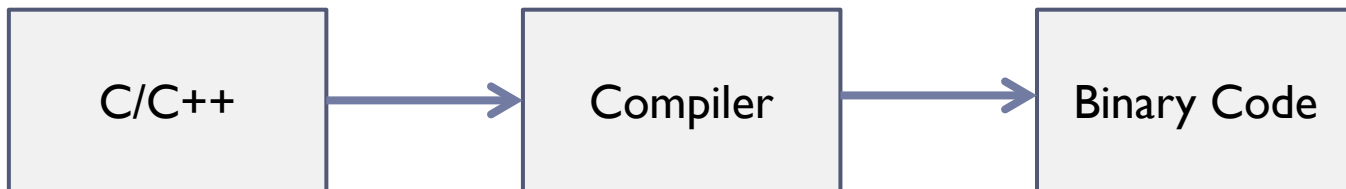
- ▶ 우리가 이해하기 쉬운
- ▶ C/C++

```
1: int x = 2;
2: while(x <= p-1){
3:     if(p % x == 0){
4:         printf("It is NOT prime");
5:         return false; // exit
6:     }
7: }
8: printf("It is prime");
9: return true;
```



Programming

- ▶ 컴퓨터는 2진수밖에 모름
 - ▶ 근데 어떻게 C/C++과 같이 Human-friend Language를 Computer가 이해할 수 있을까?
 - ▶ **Compiler**



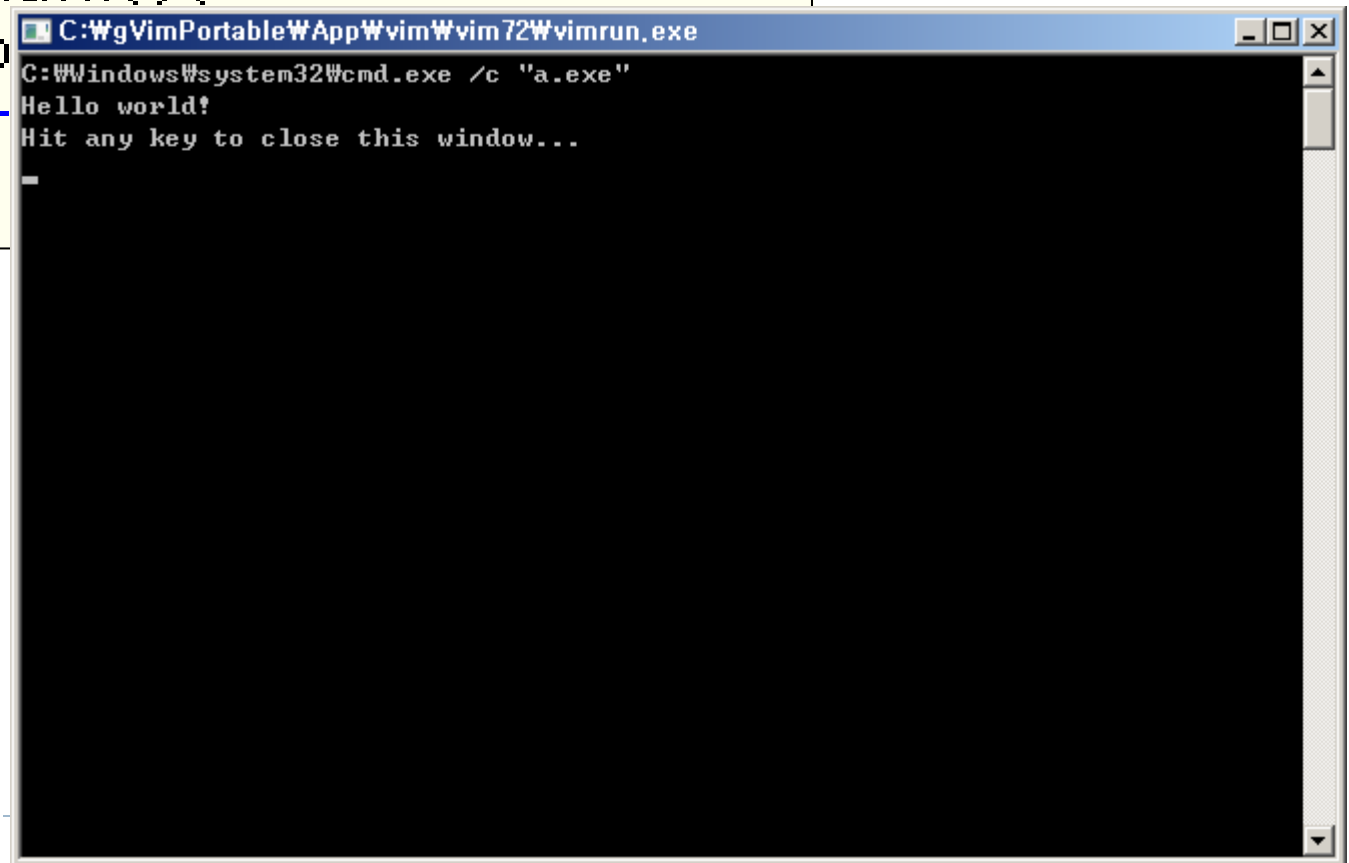
Hello world!

```
1: #include <stdio.h>
2: int main(){
3:     printf("Hello world!");
4:     return 0;
5: }
```



Hello world!

```
1: #include <stdio.h>
2: int main(){
3:     p
4:     r
5: }
```



C:\WgVimPortableWAppWvimWvim72Wvimrun.exe
C:\Windows\system32\cmd.exe /c "a.exe"
Hello world!
Hit any key to close this window...

Contents

Data type with I/O

Function

Conditional statement

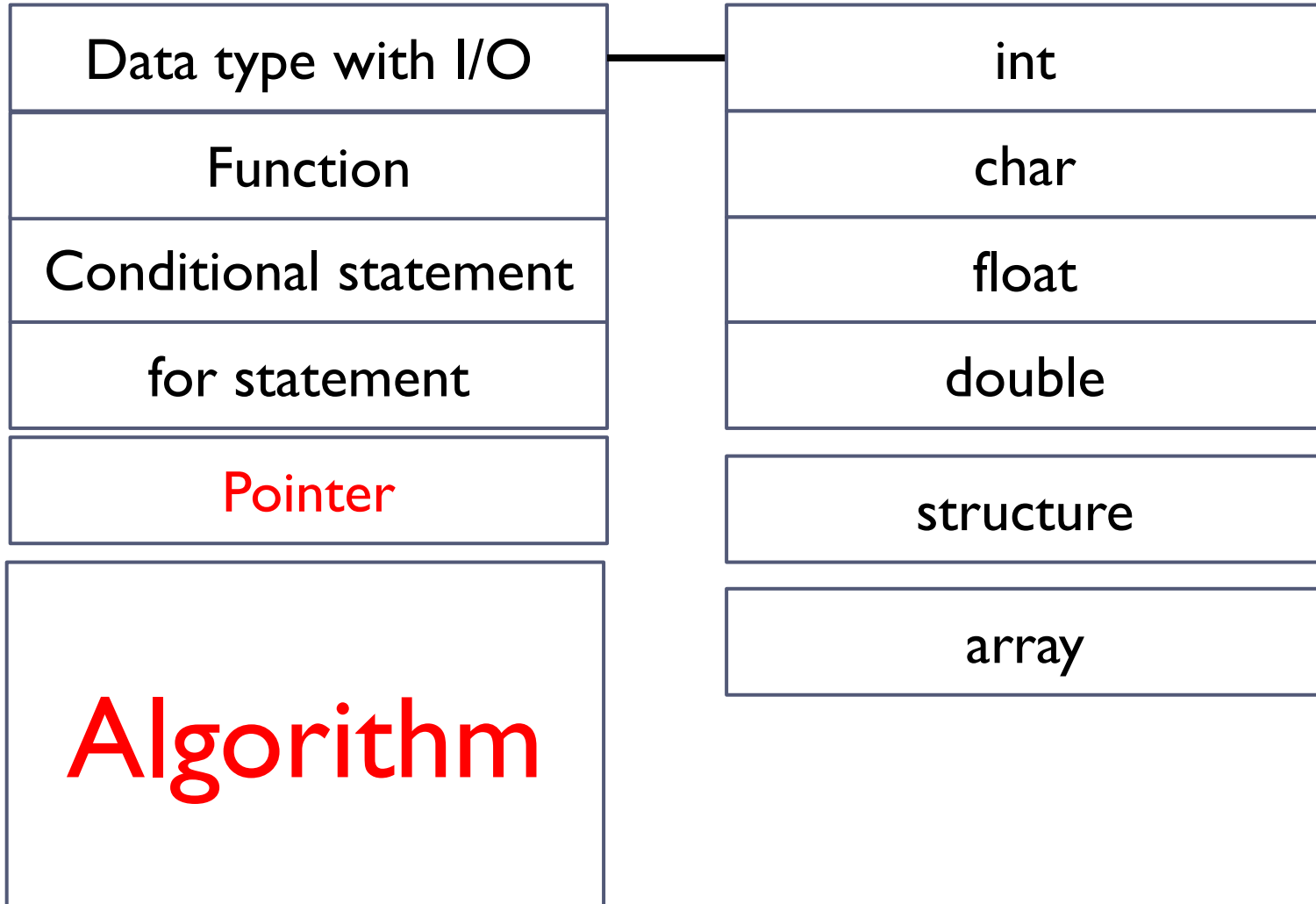
for statement

Pointer

Algorithm



Contents



Variable

- ▶ 값을 담을 수 있는 주머니
 - ▶ int : 정수
 - ▶ float : 실수
 - ▶ double : float보다 큰 실수
 - ▶ char : 문자

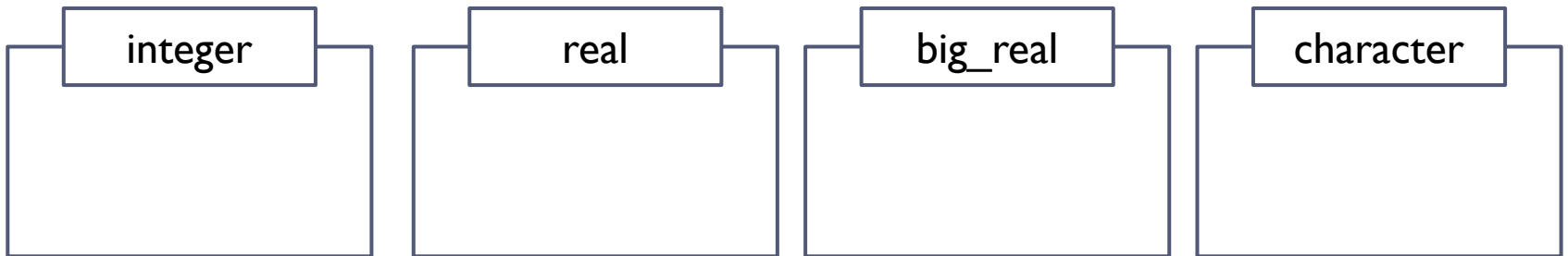
```
#include <stdio.h>
int main(){
    int integer;
    float real;
    double big_real;
    char character;
    return 0;
}
```



Variable

- ▶ 값을 담을 수 있는 주머니
 - ▶ int : 정수
 - ▶ float : 실수
 - ▶ double : float보다 큰 실수
 - ▶ char : 문자

```
#include <stdio.h>
int main(){
    int integer;
    float real;
    double big_real;
    char character;
    return 0;
}
```



Operator

- ▶ 단순한 연산을 할 수 있음

```
#include <stdio.h>
int main(){
    int one, two;
    int plus, minus, multiple, quotient, remain;
    one = 2;
    two = 3;
    plus = one + two;
    minus = one - two;
    multiple = one * two;
    quotient = one / two;
    remain = one % two;
    return 0;
}
```



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ `x = 3.4;` 라고 하면 정수부분만 나와서 `x`에는 3이 assign

```
#include <stdio.h>
int main(){
    int x;
    x = 1823781231782;
    return 0;
}
```



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4;$ 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4$; 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



1 bit



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4$; 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



1 byte (= 8 bit)



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4$; 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



4 byte



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4$; 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



4 byte

1 bit에는 0 or 1 의 자료가 들어갈 수 있음.



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4$; 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



4 byte

4byte \rightarrow 32bit 이므로 2^{32} 가지의 경우의 수가 있다.



Integer

▶ 정수형

- ▶ 정수만을 담을 수 있음
- ▶ $x = 3.4$; 라고 하면 정수부분만 나와서 x 에는 3이 assign
- ▶ $-2^{31} \sim 2^{31} - 1$ 의 범위에 있는 값만 담을 수 있음
- ▶ 4 byte



4 byte

그래서 $-2^{31} \sim 2^{31} - 1$ 의 2^{32} 개의 숫자를 표현한다.



Print

- ▶ printf

- ▶ printf(“hello world!”);
- ▶ 특정 Variable이 갖고 있는 값을 출력할 수도 있음

```
printf ( “a = %d, b = %d”,a, b);
```



Print

▶ printf

- ▶ printf(“hello world!”);
- ▶ 특정 Variable이 갖고 있는 값을 출력할 수도 있음

```
printf ( “a = %d, b = %d”, a, b);
```



Print

▶ printf

- ▶ printf(“hello world!”);
- ▶ 특정 Variable이 갖고 있는 값을 출력할 수도 있음

```
printf ( “a = %d, b = %d”, a, b );
```



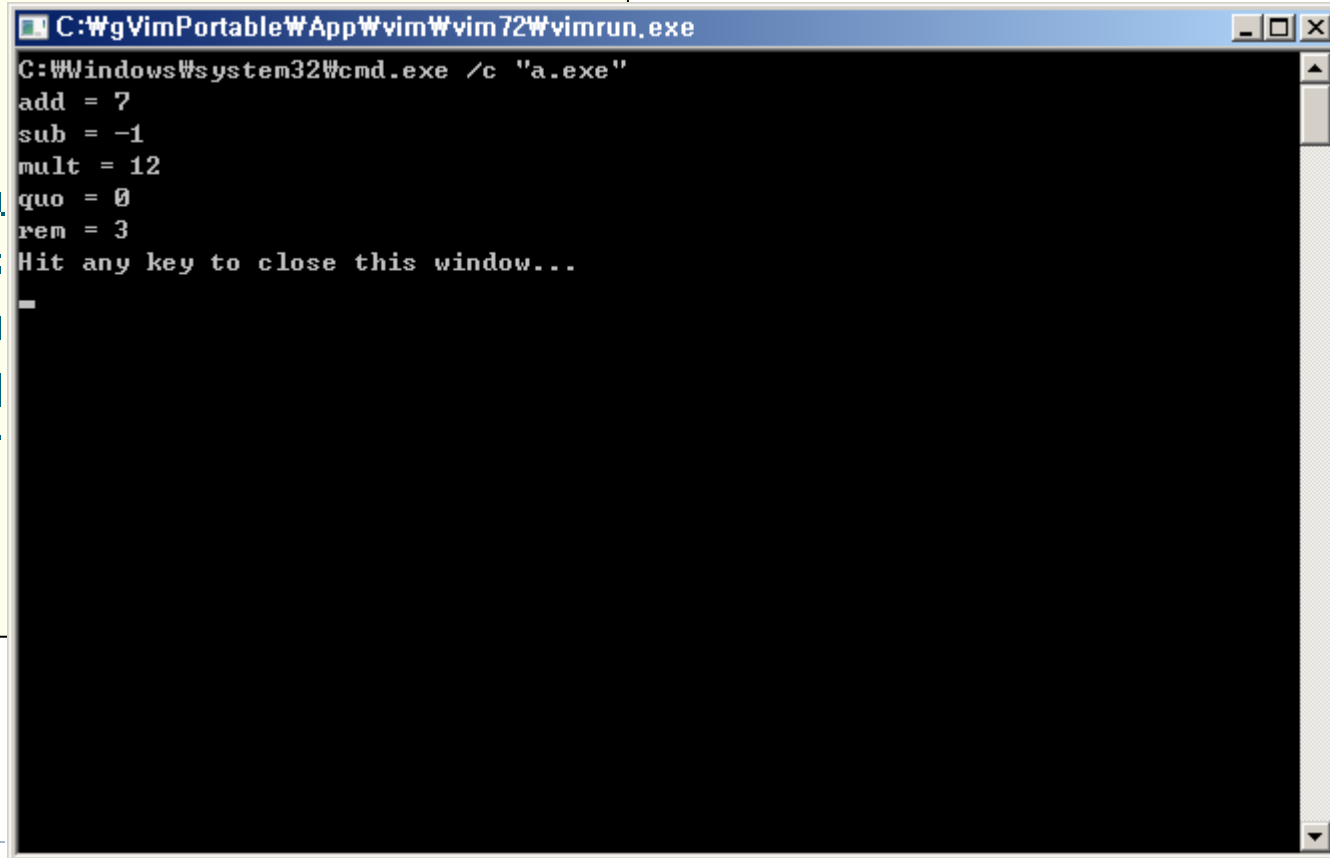
Print

```
#include <stdio.h>
int main(){
    int x, y;
    x = 3;
    y = 4;
    printf("add = %d\n", x+y);
    printf("sub = %d\n", x-y);
    printf("mult = %d\n", x*y);
    printf("quo = %d\n", x/y);
    printf("rem = %d\n", x%y);
    return 0;
}
```



Print

```
#include <stdio.h>
int main(){
    int x, y;
    x = 3;
    y = 4;
    printf("a
printf("s
printf("m
printf("q
printf("r
    return 0;
}
```



```
C:\WgVimPortable\App\Vim\vim72\vimrun.exe
C:\Windows\system32\cmd.exe /c "a.exe"
add = 7
sub = -1
mult = 12
quo = 0
rem = 3
Hit any key to close this window...
-
```