Basis and Practice in Programming
week3
Data type of Variables

- Many types in C language
  - Integer, Character, Floating-point ...

- int
  - Integer type variable
  - 4 bytes of memory space

- char
  - Character type variable
  - 1 byte of memory space

- float
  - Real number type variable
  - 4 bytes of memory space

- Double
  - For big size variable
  - 8 bytes of memory space
Modifier of Variables

- \%d / \%u: Signed integer / Unsigned integer
- \%f: Signed floating point number
- \%.nf: limited number under the zero
- \%s: String
- \%c: Single character
- \%%: printing ‘%’
• Declaration
  – Every variables must be declared before using
  – Usage: `<keyword> <name of variable>`

```c
/* Week 3
 * Declare variables */
#include <stdio.h>

int main(void)
{
    a = 10;
    printf("a = %d\n", a);
    return 0;
}
```
• Declaration
  – Every variables must be declared before using
  – Usage : <keyword> <name of variable>

```c
/* Week 3
 * Declare variables */
#include <stdio.h>

int main(void)
{
    a = 10;
    printf("a = %d\n", a);
    return 0;
}
```

```c
/* Week 2 practice 3
 * Declare variables */
#include <stdio.h>

int main(void)
{
    int a;
    a = 10;
    printf("a = %d\n", a);
    return 0;
}
```
Name of Variables (1/2)

• Composed of Alphabet, number and under-bar(_)

• Capital letter is not equal to small letter
  – int a and int A are different variable
  – Int (x) int (o), float (o) Float (x)

• Can not use numbers at the first of variables
  – int a1 (o), int 1a (x)

• Special keyword and space character are not allowable
/* Week 3  
 * Name of variables */
#include <stdio.h>

int main(void)
{
    int a;
    int A;
    int int_a; //correct
    //float int //wrong
    char c, d; //declaration of two variables at the same time

    return 0;
}
Character Representation (1/2)

- Character representation in C
  - Just integer number
  - ASCII: American Standard Code for Information Interchange

```c
#include <stdio.h>

int main(void)
{
    char sa = 'a', sb = 'b';
    char la = 'A', lb = 'B';

    printf("%c %c %c %c\n", sa, sb, la, lb);
    printf("%d %d %d %d\n", sa, sb, la, lb);

    return 0;
}
```

**Printing format**
- `%c`: character format
- `%d`: integer format
• Character representation in C (continued)

```c
#include <stdio.h>

int main(void)
{
    char foo = 48, bar = 97;
    int i = 0;

    while (i < 10) {
        foo += 1;
        bar += 1;
        printf("%c %c\n", foo, bar);
        i++;
    }

    return 0;
}
```

ASCII Table:
http://www.ascii-table.com/
Integer Representation (1/3)

• Integer representation in C
  – Char type: 8 bits (1 byte)
  – Int type: 32 bits (4 bytes)
  – Sign bit: representing +/-, Value bits: value of number

![Diagram showing binary representation of char and int types]

- Char: 8 bits
- Int: 32 bits
- MSB: sign bit
- Value bits
#include <stdio.h>

int main(void)
{
    int num = 0xffffffff; //maximum value in int
    printf("%d %u\n", num, num);
    return 0;
}

**Printing format**
%u : unsigned integer format
### scanf() and printf() (1/3)

- **Input/Output function**
  - Declared in standard input output header file (`stdio.h`)

- **Formatted scan function : scanf**
  - **Usage** : `scanf("<input type>", &<variable>);`

- **Formatted print function : printf**
  - **Usage** : `printf("<output type>", <variable>);`

- **Input/output type (Modifier)**
  - `%d : decimal number, %c : character`
```c
/* Week 3
 * Name of variables */

#include <stdio.h>
int main(void)
{
    int result;
    int val1, val2;
    char c = 'k';

    printf("First number : ");
    scanf("%d", &val1);
    printf("Second number : ");
    scanf("%d", &val2);

    result = val1 + val2;
    printf("%c\n", c);
    printf("%d + %d = %d\n", val1, val2, result);

    return 0;
}
```
```c
/* Week 3
 * Name of variables */

#include <stdio.h>
int main(void)
{
    int result;
    int val1, val2;

    scanf("%d %d", &val1, &val2); //separated by space

    result = val1 + val2;

    printf("%d + %d = %d \n", val1, val2, result);

    return 0;
}
```
Arithmetic Operation (1/2)

- **Addition**
  - Usage: `result = a + b;`
- **Subtraction**
  - Usage: `result = a - b;`
- **Multiply**
  - Usage: `result = a * b;`
- **Division**
  - Usage: `result = a / b;`
- **Modular**
  - Usage: `result = a % b;`

```c
/* Week 3
 * Arithmetic operation */
#include <stdio.h>

int main(void)
{
    int a, b, result;
    a = 10, b = 3;

    result = a + b;
    printf("%d + %d = %dn", a, b, result);

    result = a - b;
    printf("%d - %d = %dn", a, b, result);

    result = a * b;
    printf("%d * %d = %dn", a, b, result);

    result = a / b;
    printf("%d / %d = %dn", a, b, result);

    result = a % b;
    printf("%d %% %d = %dn", a, b, result);

    return 0;
}
```
Arithmetic Operation (2/2)

```c
/* Week 3
 * Arithmetic operation */
#include <stdio.h>

int main(void)
{
    int a, b;
    a = 10, b = 3;

    a += b; // same as a = a + b
    printf("%d\n", a);

    a -= b; // same as a = a - b
    printf("%d\n", a);

    a *= b; // same as a = a * b
    printf("%d\n", a);

    a /= b; // same as a = a / b
    printf("%d\n", a);

    a %= b; // same as a = a % b
    printf("%d\n", a);

    return 0;
}
```
Exercise (1/3)

• Compile
  – gcc <filename>.c -o <output>
  – $gcc practice1.c -o practice1

• Make output file
  – $./practice1 < input_ex1.txt > output_ex1.txt

• Diff command
  – You must check your output by sample output
  – $diff output_ex1.txt sample_output.txt
  – If any message isn’t printed, submit your code to I-campus
Exercise(2/3)

- Week3 exercise 1
  - Simple arithmetic operating program
  - Enter two numbers
  - Print each results of operation ‘+’, ‘-’, ‘*’, ‘/’ and ‘%’ at new line
  - All outputs must be INTEGER format
  - Due : Class time

<table>
<thead>
<tr>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
<tr>
<td>-2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Week3 exercise 2
- Simple arithmetic operating program
- Enter two numbers
- Print each results of operation ’+’, ’-’, ’*’, ’/’ at new line
- All outputs must be FLOAT format
- Due : Class time
Exercise(3/3)

- Week3 exercise 2
  - Simple arithmetic operating program
  - Enter two numbers
  - Print each results of operation ‘+’, ‘-’, ‘*’, ‘/’ at new line
  - All outputs must be FLOAT format
  - Due: Class time

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
</table>
| 1 3   | 4.000000
|       | -2.000000
|       | 3.000000
|       | 0.333333