

# C Programming

# Course Basics

## Instructor

- [Joonwon Lee](#)
  - joonwon at skku edu
  - 031 299 4592
  - Semiconductor Bldg 400626
- TA
  - [Sang-Wook Kim](#)
  - [Jong-Won Kim](#)

## Lectures

- class 45 – Monday 13:00 ~
- class 46 – Tuesday 13:00 ~
  - both at 400118
- two hours lectures
- 2~3 hours lab at 400202

# Course Materials

- Textbook

Kelley A., Pohl I, "[A Book on C: Programming in C](#)", Fourth Edition, Addison-Wesley, 1998, ISBN 0-201-18399-4.

<http://www.cs.ucsc.edu/~pohl/abc4.html>

- Course Web - <http://csl.skku.edu/GEDD0072010F/GEDD0072010F>

- Laboratory

- conducted by TA

- lectures and programming exercises

- Homeworks

- 5 individual homeworks

# Academic Honesty

- All work submitted for credit must be your original ones.
- Cheating on lab or homework
  - “F” grade and a report to the department
- Cheating on examination
  - report to the president of SKKU

# An Introduction to C

- 1972: developed by Dennis Ritchie
  - to develop an OS(Unix) for PDP-11
  - small
  - efficient
- 1989: ANSI C
  - portable
- Java is slow
  - JVM, instead of CPU, runs bytecode
  - string, vector, class, .....

# C, C++, Java

- Java is safe and elegant, but slow
- C++ is unsafe and fast, also complex
- C is unsafe, but fast and simple
  - a small language (not many features)
  - portable
  - modular
  - basis for C++ and Java

- # for preprocessor
- indicates where to look for printf() function
- .h file is a header file

```
#include <stdio.h>
```

- entry point
- called on program start
- only one main( ) in any program

```
int main(void)
```

```
{
```

```
    printf("Hello, world!\n");
```

```
    return 0;
```

```
}
```

- belongs to stdio.h
- "Hello...." is a parameter to printf()

# Marathon Distance Program

- convert the distance to kilometers
  - 1 mile = 1.609 km = 1760 yards
  - we know that the marathon length is 26 miles and 385 yards, then what is it in kilometers?
    - the answer is 42.185968



```
/* The distance of a marathon in kilometers. */
```

**comment**

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int    miles, yards;  
    float  kilometers;
```

**declaration of  
variables**

```
    miles = 26;
```

```
    yards = 385;
```

**assignment**

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf("\nA marathon is %f kilometers.\n\n", kilometers);
```

```
    return 0;
```

**expression**

```
}
```

# Preprocessor

- performs before compilation
- # indicates that this line is a directive
- #define for symbolic constants

```
#define PI 3.141592
#define YARDS_PER_MILE 1760
```
- #include <file-name> imports a header file from some where
- #include "file-name" from your directory

```
#include <stdio.h>
```

```
#define AREA 2337
```

```
#define SQ_MILES_PER_SQ_KILOMETER  
0.3861021585424458
```

```
#define SQ_FEET_PER_SQ_MILE (5280 * 5280)
```

```
#define SQ_INCHES_PER_SQ FOOT 144
```

```
#define ACRES_PER_SQ_MILE 640
```

pacific\_sea.h

```
/* Measuring the Pacific Sea. */
```

```
#include "pacific_sea.h"
```

```
int main(void)
```

```
{
```

```
    const int    pacific_sea = AREA;    /* in sq kilometers */
```

```
    double      acres, sq_miles, sq_feet, sq_inches;
```

```
    printf("\n\nThe Pacific Sea covers an area");
```

```
    printf(" of %d square kilometers\n", pacific_sea);
```

pacific\_sea.c

# I/O Using `stdio.h`

- `printf("any string or characters %d %f", a, b);`
  - " " indicates a format to be displayed
  - % is followed by a single character for a format
    - c (char), d (decimal), e (exponential), f(floating), s (string)
  - escape with `\`
    - `\n`, `\t`, `\"`, `\\`
- `scanf("%d", &age);`
  - takes something from the standard input, and interpret as a decimal

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    char    c1, c2, c3;
```

```
    int     i;
```

```
    float   x;
```

```
    double  y;
```

```
    printf("\n%s\n%s", "Input three characters,"  
           "an int, a float, and a double: ");
```

```
    scanf("%c%c%c%d%f%lf", &c1, &c2, &c3, &i, &x, &y);
```

```
    printf("\nHere is the data that you typed in:\n");
```

```
    printf("%3c%3c%3c%5d%17e%17e\n\n", c1, c2, c3, i, x, y);
```

```
    return 0;
```

```
}
```

# Control Flow

- each statement is executed one by one sequentially
- special statements change the flow
  - **if** (expr) a single statement OR { statements }
  - **while** (expr) a single statement OR
  - **for** (expr1; expr2; expr3) a single statement OR

```
expr1;  
while (expr2) {  
    statement  
    expr3;  
}
```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int i = 1, sum = 0;
```

```
    while (i <= 5) {
```

```
        sum += i;
```

```
        ++i;
```

```
    }
```

```
    printf("sum = %d\n", sum);
```

```
    return 0;
```

```
}
```

# Arrays

- deal with multiple same type data
- `int xxx[3];`
  - declares 3 integers; `xxx[0]`, `xxx[1]`, `xxx[2]`

```
int i;
```

```
i = 2;
```

```
xxx[i] = xxx[0] + 79;
```



- a string "abc"





# Pointer

- address is a location in the imaginary space
  - an array name

```
int age[100];
```

```
char *p;
```

```
int *pq;
```

# Functions

- Can you write a program of 10,000 lines in a single file?
  - divide your whole code into many small chunks
  - some chunks may look similar
    - make them into a single one; how?
    - this is a function
- `main()` is a special function called by ....

```

#include <stdio.h>

float  maximum(float x, float y);
float  minimum(float x, float y);
void   prn_info(void);

int  main(void)
{
    int    i, n;
    float  max, min, x;

    prn_info();
    printf("Input n: ");
    scanf("%d", &n);
    printf("\nInput %d numbers:", n);
    scanf("%f", &x);
    max = min = x;
    for (i = 2; i <= n; ++i) {
        scanf("%f", &x);
        max = maximum(max, x);
        min = minimum(min, x);
    }
}

```

```

float  maximum(float x, float y)
{
    if (x > y)
        return x;
    else
        return y;
}

```

```

float  minimum(float x, float y)
{
    if (x < y)
        return x;
    else
        return y;
}

```

```

void  prn_info(void)
{
    printf("\n%s\n%s\n\n",
        "This program reads an integer value
for n, and then",
        "processes n real numbers to find
max and min values.");
}

```

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    int    a = 1;
```

```
    void   try_to_change_it(int);
```

```
    printf("%d\n", a);    /* 1 is printed */
```

```
    try_to_change_it(a);
```

```
    printf("%d\n", a);    /* 1 is printed again! */
```

```
    return 0;
```

```
}
```

```
void try_to_change_it(int a)
```

```
{
```

```
    a = 777;
```

```
}
```

# Files

- you need files, believe me.
- `xfp = fopen("file-name", "r");`
  - checks if the file is available
  - prepares a pointer, `xfp`, to a location inside a file
- now read from (write to) the file using the pointer

```
c = getc(xfp);
```

```
n = fscanf, "%d %d %f %s", i, j, x, name);
```

```
/* Count uppercase letters in a file. */
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    int    c, i, letter[26];
```

```
    FILE   *ifp, *ofp;
```

```
    ifp = fopen(argv[1], "r");
```

```
    ofp = fopen(argv[2], "w");
```

```
    for (i = 0; i < 26; ++i)    /* initialize array to zero */
```

```
        letter[i] = 0;
```

```
    while ((c = getc(ifp)) != EOF)
```

```
        if (c >= 'A' && c <= 'Z')    /* find uppercase letters */
```

```
            ++letter[c - 'A'];
```

```
    for (i = 0; i < 26; ++i) {        /* print results */
```

```
        if (i % 6 == 0)
```

```
            putc('\n', ofp);
```

```
            fprintf(ofp, "%c:%5d    ", 'A' + i, letter[i]);
```

```
    }
```

```
    putc('\n', ofp);
```

```
    return 0;
```

```
}
```