w12-Lab

Function Part II

Assembled for 204111

by Kittipitch Kuptavanich

Flow of Execution

- โดยปกติแล้ว ในโปรแกรมใด ๆ ลำดับการ **execute** จะเริ่มจากบนสุดของโปรแกรม ทีละ **statement** จนถึงส่วนล่างสุดของโปรแกรม
- การสร้าง **function** ขึ้นมาไม่ได้เปลี่ยนแปลงลำดับการ **execute**
- **Statement** ใด ๆ ที่ **define** ขึ้นใน **function** จะไม่ถูก **execute** จนกว่า **function** นั้น ๆ จะถูกเรียกใช้

Flow of Execution

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_STUDENT 100

// prototypes
void readInScore(double score[]);
double calAverage(double score[]);
double calSd(double score[],double mean);

int main()
{
   double score[MAX_STUDENT];

   readInScore(score);

   double mean = calAverage(score);

   double sd = calSd(score,mean);

   printf("mean = %.2f\nsd = %f \n", mean,sd);

   return 0;
}
```


Flow of Execution [2]

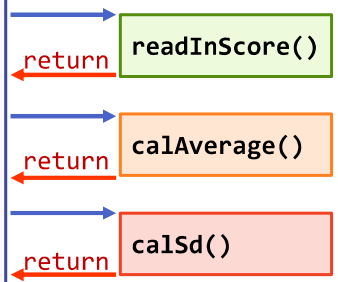
```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_STUDENT 100

// prototypes
void readInScore(double score[]);
double calAverage(double score[]);
double calSd(double score[],double mean);

int main()
{



}
```



return Statement

```
int functionX(......) {

    if (z == a) {
        return a;
    }

    while (.......) {
        ....
        ....
        if (y != b)
            return y;
    }
    ...
    ...
    if (x < a)
        return x;
    else
        return y;
}
```

- ใน function x ใด ๆ หากมีการดำเนินการมาถึง return statement (แม้จะมี return type เป็น void) function จะหยุดทำการทันที
- การ execute จะกลับไปที่ function ที่เรียกใช้ function x
- กรณีไม่มี return statement (void) จะดำเนินการถึง end of block (เครื่องหมายปีกกา })

Type Casting on Function Calls

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

// prototypes
int averageOf2(int num1, int num2);

int main()
{
    double num1 = 3.8;
    double num2 = 4.9;
    double result = averageOf2(num1, num2);
    printf("mean of %.2f and %.2f = %.2f\n",result);

    return 0;
}

int averageOf2(int num1, int num2) {


    double result = (num1 + num2) / 2;

    return result;
}
```

```
mean of 3.80 and 4.90 is 3.00
```


Type Casting on Function Calls [2]

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

// prototypes
int averageOf2(int num1, int num2);

int main()
{
    double num1 = 3.8;
    double num2 = 4.9;
    double result = averageOf2(num1, num2);
    printf("mean of %.2f and %.2f = %.2f\n",result);

    return 0;
}

int averageOf2(int num1, int num2) {

    printf("%d %d\n",num1,num2);
    double result = (num1 + num2) / 2;
    printf("%f\n",result);
    return result;
}
```

Implicit casting!!!

Variable Scope

- โดยทั่วไปแล้ว variable ที่ถูกสร้างขึ้นภายใน block (เครื่องหมายปีกกา {}) ใด ๆ จะสามารถอ้างถึง (read/write) ได้ ภายใน block นั้น ๆ
- เช่นเดียวกันกับกรณี function
  - Variable ที่ถูกสร้างขึ้นใน function จะสามารถอ้างถึงได้ใน function นั้น ๆ เท่านั้น
- Variable scope คือ block หรือ section ของโปรแกรมที่ variable ใด ๆ สามารถถูกอ้างถึงได้

Variable Scope [2]

- **Local variable** หรือ variable ที่มี **local scope** จะมีการจองพื้นที่ด้วยการประกาศตัวแปร ภายใน function body (ระหว่างปีกกา)

```
int main() {
  int x = 3;
  …
}
```

- **Global variable** หรือ variable ที่มี **global scope** จะมีการจองพื้นที่ด้วยการประกาศตัวแปรนอก function ใด ๆ

Variable Scope [3]

For 204111

(And maybe the rest of your student life)

Global Constant: OK

Global Variable: NO!!!!

(unless you have a really good reason)

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

const int DAYS_IN_WEEK = 7;  // Global constant Variable: OK
int num = 5;                 // Global Variable: NOT recommended

// prototypes


int main()
{
    num++;                   // Modifying Global Variable
    someFunction(5);
    return 0;
}

void someFunction(int num1) {
    int y = 3;               // Local Variable
    num += num1;             // Modifying Global Variable

}
```


Function Calling with Arrays

- ในภาษา C เราสามารถ ส่งผ่าน array เป็นค่า parameter ไปยัง function ได้ดังนี้

```
1. void printArray(char s1[],…)
2. void printArray(char s1[30],…)
3. void printArray(char *s1,…)        //204112
```

- ทั้งสามวิธีเป็นการส่ง array ไปยัง function ในรูปแบบการส่ง memory address ทั้งสิ้น

Function Calling with Arrays [2]

- เนื่องจากภาษา C ไม่มีกลไกในการ check ขอบเขตของ array, หากเขียน function ขึ้นเองควรต้อง pass ขนาด array เมื่อมีการ เรียกใช้ function ด้วย เช่น
  - `void printArray(char *s1, int size,…);`

Function Calling with Arrays [3]

```
void printArray(char* a1, int size) {

  int i;
  for(i=0; i< size; i++)
  ....
}
```

- โดยเมื่อมีการเรียกใช้ เราสามารถหาขนาด array `a1` (จำนวน block) ได้ดังนี้
- `printArray(a1, sizeof(a1)/sizeof(a1[0]));`
- ข้อควรระวัง: การเรียกใช้ `sizeof(a1)` ภายใน function `printArray()` จะได้ขนาดของ address แทน (= 4 byte) [-_-] ไม่ใช่ขนาดของ array a1. [ไม่ควรทำ]

Practice 1

- ให้เขียนโปรแกรมเพื่อรับค่าจำนวนเต็มไม่เกิน 2 หลัก แล้วแสดงคำอ่านในภาษาอังกฤษ โดยจะต้องมีการสร้าง user defined function ดังนี้
- `void twoDigitsToWord(int num)`

ตัวอย่างการ run 1

```
Enter a number: 25
English Words: twenty five
```

ตัวอย่างการ run 2

```
Enter a number: 19
English Words: nineteen
```


Practice 1 [2]

Problem Solving

- แยกเป็น 2 case:
  - 1 – 19: *one*, *two*, *three*,…. *nineteen*
  - 20 – 99: twenty, twenty *one*, twenty *two*…..
    thirty, thirty *one*, thirty *two*…..

Practice 1 [3]

```
void digit10sToWord(int num) // หลักสิบ
{

    switch(num) {
    case 2:
        printf("twenty");
        break;
    case 3:
        printf("thirty");
        break;
    ...
    ...
    case 8:
        printf("eighty");
        break;
    case 9:
        printf("ninety");
        break;
    }

}
```

```
void unitToWord(int num)
{
    switch (num) {
    case 1:
        printf("one");
        break;
    case 2:
        printf("two");
        break;
    ...
    case 18:
        printf("eighteen");
        break;
    case 19:
        printf("nineteen");
        break;

    }
```

Practice 1 [4]

```
void twoDigitsToWord(int num) {
    int tens;
    int unit;

    if (num > 19) {



    } else {


    }
    printf( . . .);



}
```