344-111

FUNCTIONS

Outline

- Function Definition
- Type of function
- Passing structure to Function
- Recursive

Defining a function

```
type function_name(type arg1, type arg2. ...)
{
    local variable declaration;
    statement1;
    statement2;
    ...
    statement n;
    return(value);
}
```

Type: ชนิดของฟังก์ชัน
Function_name: ชื่อฟังก์ชัน
Type arg1 : กำหนดชนิดอาร์กิวเมนต์ที่ส่งให้กับฟังก์ชัน
Local variable: ตัวแปรสำหรับใช้งานในฟังก์ชัน
return(value) : ส่งผลลัพธ์การทำงานของฟังก์ชัน กลับไปยังคำสั่งที่เรียกใช้ฟังก์ชัน โดยผลลัพธ์จะต้องเป็นชนิดเดียวกับชนิดของฟังก์ชันที่ระบุ

Type of Function

- พิจารณาได้
  - ฟังก์ชันที่ไม่มีการรับค่า/ส่งค่า
  - ฟังก์ชันที่มีการรับค่า
    - Pass-By-Value means the actual value is passed on
    - Pass-By-Reference means a number (called an address) is passed on which defines where the value is stored

ไม่มีการรับค่า/ส่งค่า

- No arguments sent to a function
- No return value from a function
- Example

```
1   #include<stdio.h>
2
3   void message()
4   {
5       printf("Hello!");
6   }
7   main()
8   {
9       message();
10  }
```

```
Hello!
```

มีการรับค่า

- มี 2 แบบ
  - Pass by value
    - รับค่าของตัวแปร ค่าคงที่ หรือนิพจน์
    - ให้ผลออกมาเป็นค่าคงที่
    - ถ้าตัวแปรตรงกันระหว่างฝ่ายเรียกและฝ่ายถูกใช้งานถือว่าคนละตัวกัน
  - Pass by reference
- การคืนค่า
  - ไม่มีการคืนค่าใช้ชนิดของฟังก์ชันเป็น void
  - มีการคืนค่า

Pass-By-Value

- Passing by value means that the value of the function parameter is copied into another location of your memory
- Accessing or modifying the variable within your function, only the copy is accessed/modified and the original value is left untouched
- ตัวอย่าง

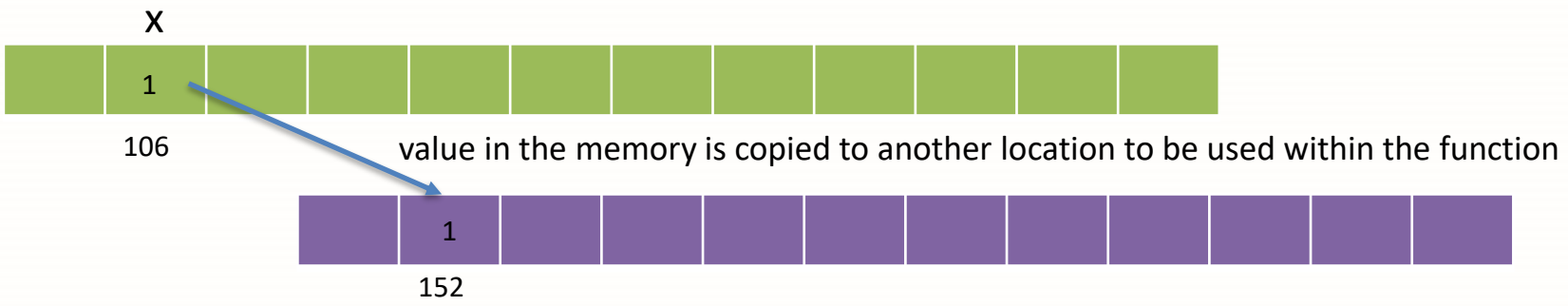
```
1   #include<stdio.h>
2   int increaseByOne(int);
3   void main()
4   {
5       int x = 1;
6       printf("main: value x before call increaseByOne %d\n\n",x);
7       int xx = increaseByOne(x);
8
9       printf("main: value x after call increaseByOne %d\n\n", x);
10  }
11
12  int increaseByOne(int x)
13  {
14      x = x+1;
15      printf("main: value x before call increaseByOne %d\n\n",x);
16      return x;
17  }
```

Variable x is passed to function

Function can use x

How memory works

```
1   #include<stdio.h>
2   int increaseByOne(int);
3   void main()
4   {
5       int x = 1;
6       printf("main: value x before call increaseByOne %d\n\n",x);
7       int xx = increaseByOne(x);
8
9       printf("main: value x after call increaseByOne %d\n\n", x);
10  }
11
12  int increaseByOne(int x)
13  {
14      x = x+1;
15      printf("main: value x before call increaseByOne %d\n\n",x);
16      return x;
17  }
```

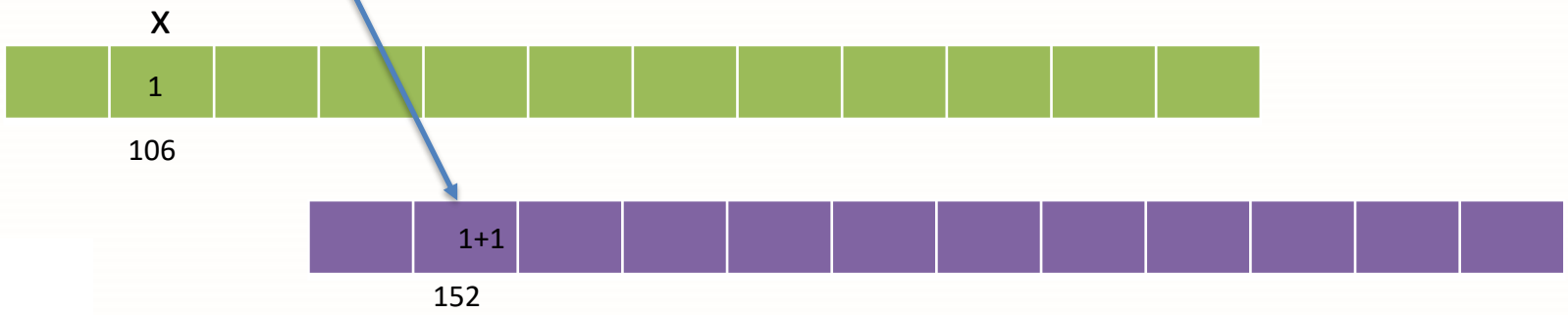


modify/update

```c
#include<stdio.h>
int increaseByOne(int);
void main()
{
    int x = 1;
    printf("main: value x before call increaseByOne %d\n\n",x);
    int xx = increaseByOne(x);

    printf("main: value x after call increaseByOne %d\n\n", x);
}

int increaseByOne(int x)
{
    x = x+1;
    printf("main: value x before call increaseByOne %d\n\n",x);
    return x;
}
```

```
main: value x before call increaseByOne 1

main: value x before call increaseByOne 2

main: value x after call increaseByOne 1
```

Only one value can change by returning



Pass-by Value

- prototype

```
type function_name(type)
```

- รูปแบบการเรียกใช้งาน

```
xx = function_name(xx)
```

Pass-by Reference

- The memory address of the variable (a pointer to the memory location) is passed to the function
- ฟังก์ชันที่ถูกเรียกใช้จะมีตัวแปรแบบพอยน์เตอร์มารับค่าตำแหน่งที่อยู่
- การดำเนินการกับตัวแปรที่รับค่าไป มีผลกับค่าของตัวแปรที่ส่งให้ฟังก์ชัน
- ตัวอย่าง

```
1  #include<stdio.h>
2  void increaseByOne(int*);
3  void main()
4  {
5      int x =1;
6      printf("main: value x before call increaseByOne %d",x);
7      increaseByOne(&x);
8      printf("main: value x after call increaseByOne %d", x);
9  }
10 void increaseByOne(int *x)
11 {
12     *x = *x+1;
13     printf("increaseByOne: %d",*x);
14 }
```

```
main: value x before call increaseByOne 1
increaseByOne: 2
main: value x after call increaseByOne 2
```

How memory works

```
1  #include<stdio.h>
2  void increaseByOne(int*);
3  void main()
4  {
5      int x =1;
6      printf("main: value x before call increaseByOne %d",x);
7      increaseByOne(&x);
8      printf("main: value x after call increaseByOne %d", x);
9  }
10 void increaseByOne(int *x)
11 {
12     *x = *x+1;
13     printf("increaseByOne: %d",*x);
14 }
```

the memory address of x is *106*

the memory address of x is sent to the function

the variable x used within the function still points to the same memory address of x in main

Pass-by Reference

- Prototype

```
void increaseByOne(int*)
```

- รูปแบบการเรียกใช้งาน

```
increaseByOne(&x)
```

Pass-by Reference

- สามารถใช้พอยน์เตอร์ เพื่อ return ค่าได้หลายๆ ค่า

```
1  #include<stdio.h>
2  void cal_circle(float radius, float *area, float *circum);
3  void main()
4  {
5      float r = 10;
6      float ans1, ans2;
7      cal_circle(r, &ans1, &ans2);
8      printf("Circle: r = %.2f has area = %.2f, circumference = %.2f\n", r, ans1, ans2);
9  }
10
11 void cal_circle(float radius, float *area, float *circum)
12 {
13     float a, c;
14     a = 3.14*radius*radius;
15     c = 2*3.14*radius;
16     *area = a;
17     *circum = c;
18 }
```

possible to change more than one variable given in the function parameter

ผลลัพธ์

```
Circle: r = 10.00 has area = 314.00, circumference = 62.80
```

Example 1

- โปรแกรมเพื่อรับข้อมูลจำนวนเต็มจากผู้ใช้ 1 จำนวน

By value

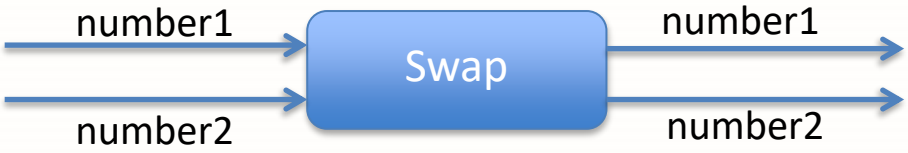
```
1  #include <stdio.h>
2  int readData();
3  void main()
4  {
5      int x;
6      x= readData();
7  }
8  int readData()
9  {
10     int data;
11     printf("Enter value  : ");
12     scanf("%d", &data);
13     return data;
14 }
```

By reference

```
1  #include  <stdio.h>
2  void  readData(int  *);
3  void  main()
4  {
5      int x;
6      readData(&x);
7  }
8  void readData(int *data)
9  {
10     printf("Enter value : ");
11     scanf("%d",data);
12 }
```

Example 2

- โปรแกรมการสลับค่าตัวแปร 2 ตัวโดยผ่านฟังก์ชัน




```
1   #include<stdio.h>
2   void swap(int *, int *);
3   void main()
4   {
5         int  x = 5,  y = 10;
6         printf("Before swap : x = %d  y = %d\n", x, y);
7         swap(&x, &y);
8         printf("After swap : x = %d  y = %d\n", x, y);
9   }
10  void  swap (int  *px,  int  *py)
11  {
12        int  temp;
13        temp = *px;
14        *px = *py;
15        *py = temp;
16  }
```

Assignment 1

- เขียนโปรแกรมเพื่อรับข้อมูลจำนวนจริง 3 จำนวนจากผู้ใช้และหาค่าเฉลี่ยของค่าที่รับเข้ามาทั้งหมด ให้หาว่ามีเลขจำนวนจริงที่รับเข้ามานั้นมีค่าน้อยกว่าค่าเฉลี่ยกี่จำนวน
- กำหนดให้มีฟังก์ชันต่อไปนี้และใช้ตัวแปรพอยน์เตอร์ในการส่งผ่านฟังก์ชัน
  - readData สำหรับอ่านข้อมูลเข้ามา
  - calAverage สำหรับคำนวนค่าเฉลี่ย
  - findLessAverage สำหรับหาจำนวนที่มีค่าน้อยกว่าค่าเฉลีย

ตัวอย่างผลลัพธ์

```
Enter value 1 : 15.5
Enter value 2 : 20.5
Enter value 3 : 25.5

The average = 20.500000

Less than average = 1
```

Assignment 2

- เข้า LMS2@PSU เพื่อตอบคำถามจากโปรแกรม

```
1   int funct1(char a, char b);
2   main()
3   {
4       char a = 'X';
5       char b = 'Y';
6       int i,j;
7       i=funct1(a,b);
8       printf("a=%c b=%c\n",a,b);
9   }
10
11  int funct1(char c1, char c2)
12  {
13      c1='P';
14      c2='Q';
15      return ((c1 < c2) ? c1:c2);
16  }
```

Assignment3

- เข้า LMS2@PSU เพื่อตอบคำถามจากโปรแกรม

```
1   int funct2(char *pa, char *pb);
2   main()
3   {
4       char a = 'X';
5       char b = 'Y';
6       int j;
7       j=funct1(&a,&b);
8       printf("a=%c b=%c\n",a,b);
9   }
10
11  int funct1(char *c1, char *c2)
12  {
13      *c1='P';
14      *c2='Q';
15      return ((*c1 == *c2) ? *c1:*c2);
16  }
```

Return a pointer from function

- Declare a function

  data-type *function-name(para-lists)

  data-type: type of the pointer that will be returned by the function

  function-name: name of a function

  para-lists: lists of parameters of the function

- Example

  int *getMaxData(int *, int *)

Example

```
1   #include <stdio.h>
2
3   int *getSum(int *, int *);
4
5   int main(void) {
6
7     int x = 100;
8     int y = 200;
9
10    int *sum = NULL;
11
12    sum = getSum(&x, &y);
13
14    printf("Sum value: %d\n", *sum);
15
16    return 0;
17  }
18
19  int *getSum(int *m, int *n) {
20
21      *m = *m + *n;
22
23      return m;
24  }
```

function that returns an integer pointer

Pointer variable

address of x and y to function

Output

```
Sum value: 300
```

Example

- Inside main function

x	y	sum
100	200	null
500	504	508

- Function call by passing address of x and y
- Inside getsum function

m	n
500	504
1000	1004

- Back from getsum

x	y	sum
300	200	1000
500	504	508

Example

```
8  #include <stdio.h>
9  int *returnFunc();
10 int main()
11 {
12     int *ptr=returnFunc();
13     printf("\n *ptr = %d",*ptr);
14     printf("\n *ptr = %d",*ptr);
15     printf("\n *ptr = %d",*ptr);
16     return 0;
17 }
18
19 int *returnFunc()
20 {
21     int i=10;
22     return &i;
23 }
```

Returning address of local variable

```
main.c:22:12: warning: function returns address of local variable [-Wreturn-local-addr]
Segmentation fault (core dumped)
```

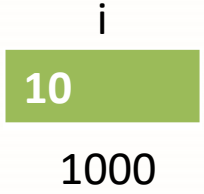
*ptr = 10
*ptr = 0
*ptr = 0

Example

- Inside main function

it is a very bad idea to return the address of a local variable to the caller function

- Inside function



- Back from a function

local variable present at address **2000** is already destroyed
ptr is pointing to the variable that does not exist

Example

- If we want to return the address of a local variable to the caller function, then it should be declared as static
- The static variable is not destroyed till the end of a program

```c
#include <stdio.h>
int *returnFunc();
int main()
{
    int *ptr=returnFunc();
    printf("\n *ptr = %d",*ptr);
    printf("\n *ptr = %d",*ptr);
    printf("\n *ptr = %d",*ptr);
    return 0;
}

int *returnFunc()
{
    static int i=10;
    return &i;
}
```

```
*ptr = 10
*ptr = 10
*ptr = 10
```

Passing structure to functions

- การส่งผ่าน structure ให้ฟังก์ชัน ทำได้ดังนี้
  - ส่งสมาชิกของ structure
  - ส่งทั้ง structure
  - ส่งอาร์เรย์ของ structure

Passing structure to functions

- การส่งค่าสมาชิกของ structure แบบ pass by value

```c
#include <stdio.h>
float cal_annual(float);
void main(void)
{
typedef struct {
    char name[20];
    int age;
    float salary;
    } employee;
    employee emp1 = {"somporn",26,10000};
    float annual;
    annual = cal_annual(emp1.salary);
    printf("Your annual salary : %.2f\n", annual);
}

float cal_annual(float sal) {
    return(sal*12);
}
```

```
Your annual salary : 120000.00
```

Passing structure to functions

- การส่งค่าสมาชิกของ structure แบบ pass by reference

```c
#include <stdio.h>
void increase_salary(float *);
void main(void) {
    typedef struct {
        char name[20];
        int age;
        float salary;
    }employee;
    employee emp1 = {"somporn",26,10000};
    float annual;
    increase_salary(&emp1.salary);
    printf("Your new salary : %.2f\n", emp1.salary);
    }

void increase_salary(float *p)
{
    (*p) *= 1.1;
}
```

```
Your new salary : 11000.00
```

Passing structure to functions

- การส่งทั้ง structure ไปให้กับฟังก์ชัน
  - Structure สามารถส่งเป็นพารามิเตอร์ของฟังก์ชัน สามารถส่งได้ทั้ง
    - pass by value
    - pass by reference
  - ส่ง array of structure ไปให้กับฟังก์ชัน
  - สามารถส่งกลับฟังก์ชันได้โดยใช้ return

Passing structure to functions

- การส่งทั้ง structure แบบ pass by value

```c
#include <stdio.h>
void printdata(employee);
typedef struct
    {
        char name[20];
        int age;
        float salary;
    }employee;
void main(void)
{
    employee emp1 = {"somporn",26,6500.50};
    float annual;
    printdata(emp1);
}
void printdata(employee one)
{
    printf("%s %d %.2f\n\n",one.name, one.age, one.salary);
}
```

```
somporn 26 6500.50
```

Passing structure to functions

- การส่งทั้ง structure แบบ pass by reference

```c
#include <stdio.h>

typedef struct
    {
        char name[20];
        int age;
        float salary;
    }employee;

void getdata(employee *p)
{
    printf("enter name: ");
    scanf("%s",p->name);
    printf("enter age: ");
    scanf("%d",&p->age);
    printf("enter salary: ");
    scanf("%f",&p->salary);
}

void main(void)
{
    employee emp1;
    getdata(&emp1);
    printf("%s %d %.2f",emp1.name,emp1.age,emp1.salary);
}
```

```
enter name: wararat
enter age: 36
enter salary: 12000
wararat 36 12000.00
```

Passing array of structure

- การส่ง array of structure ไปยังกับฟังก์ชัน
  - การ**ส่ง**ไปทั้ง array ทำได้คือ ส่ง address
  - การ**รับ**ทำได้ 2 วิธี
    - รับด้วย array
    - รับด้วย pointer

Passing array of structure

- ส่ง array แล้วรับด้วย array

```c
#include <stdio.h>
typedef struct {
    char name[40];
    int age;
    char group;
} stu;

void readdata(stu s[10], int count);
void main()
{
    int total;
    stu student[10];
    do{
        printf("Please enter number of records (Not over 10):");
        scanf(" %d", &total);
    } while(total >10);

    readdata(student, total);
    //displaydata(student, total);
}

void readdata(stu s[10], int count)
{
    int i;
    for(i=0;i<count; i++)
        {
        printf("\nRecord %d\n=============\n", i+1);
        printf("Enter name :");
        scanf(" %s", s[i].name);
        printf("Enterage :");
        scanf(" %d", &s[i].age);
        printf("Enter group:");
        scanf(" %c", &s[i].group);
        }
}
```

```
Please enter number of records (Not over 10):1

Record 1
=============
Enter name :wararat
Enterage :36
Enter group:A
```

Passing array of structure

- ส่ง array แล้วรับด้วย pointer

```c
#include <stdio.h>
typedef struct {
    char name[40];
    int age;
    char group;
} stu;

void readdata(stu *p, int count);
//void displaydata(stu *p, int count);
void main()
{
    int total;
    stu student[10];
    do{
        printf("Please enter number of records (Not over 10):");
        scanf(" %d", &total);
    }while(total >10);

    readdata(student, total);
    //displaydata(student, total);
}

void readdata(stu *p, int count) {
    int i;
    for(i=0;i<count; i++)
        { /* Print record number on screen (start with 1) */
        printf("\nRecord %d\n=============\n", i+1);
        printf("Enter name :");
        scanf(" %s", &(p+i)->name);
        printf("Enter age :");
        scanf(" %d", &(p+i)->age);
        printf("Enter group :");
        scanf(" %c", &(p+i)->group);
        }
}
```

```
Please enter number of records (Not over 10):1

Record 1
=============
Enter name :wararat
Enter age :36
Enter group :A
```

Return structure from function

- Return – type with struct

```
struct  <ชื่อโครงสร้าง>  <ชื่อฟังก์ชัน> ( <ค่าที่ส่งมายังฟังก์ชัน> ) {
        struct   <ชื่อโครงสร้าง>  <ตัวแปรโครงสร้าง>
         .....
       return ( <ตัวแปรโครงสร้าง> );
}
```

- Example

```
struct  point  makePoint ( ) {
      struct  point  p1;
      ...
      return (p1);
}
```

Example

```
3   struct student{
4           int id;
5           char name[20];
6           int age;
7           float height;
8           float weight;
9       };
10
11  struct student Input();
12
13  void main()
14  {
15      struct student std;
16      std = Input();
17      printf("\n\n%d %s %d %.2f %.2f\n\n",
18             std.id,std.name,std.age,std.height,std.weight);
19  }
20
21  struct student Input()
22  {
23      struct student E;
24
25      printf("\nEnter student Id : ");
26      scanf("%d",&E.id);
27      printf("\nEnter student Name : ");
28      scanf("%s",&E.name);
29      printf("\nEnter student Age : ");
30      scanf("%d",&E.age);
31      printf("\nEnter student height : ");
32      scanf("%ld",&E.height);
33      printf("\nEnter student height : ");
34      scanf("%ld",&E.weight);
35
36      return E;
37  }
```

```
Enter student Id : 1
Enter student Name : Jane
Enter student Age : 18
Enter student height : 163
Enter student height : 48

1 Jane 18 163.00 48.00
```

Return structure pointer from function

- Return – type with struct

```
struct <ชื่อโครงสร้าง> *<ชื่อฟังก์ชัน> ( <ค่าที่ส่งมายังฟังก์ชัน> ) {
        struct <ชื่อโครงสร้าง> <ตัวแปรโครงสร้าง>
        .....
      return ( <ตัวแปรโครงสร้าง> );
}
```

- Example

```
struct point *makePoint ( ) {
    struct point p1;
    ...
    return (p1);
}
```

Example

```
5   struct student{
6           int id;
7           char name[20];
8           int age;
9           float height;
10          float weight;
11      };
12
13  struct student *Input();
14
15  void main()
16  {
17
18      struct student *std;
19      std = Input();
20      printf("\n\n%d %s %d %.2f %.2f\n\n",
21              (*std).id,(*std).name,(*std).age,(*std).height,(*std).weight);
22  }
23
24  struct student *Input()
25  {
26      struct student *students;
27      students = malloc(sizeof(struct student));
28      students->id = 1;
29      strcpy(students->name, "Jane");
30      students->age = 18;
31      students->height = 163.0;
32      students->weight = 48.0;
33
34      return students;
35  }
```

```
1 Jane 18 163.00 48.00
```

Example

```
5  struct student{
6          int id;
7          char name[20];
8          int age;
9          float height;
10         float weight;
11     };
12
13 struct student *Input();
14
15 void main()
16 {
17     struct student *std;
18     std = Input();
19     printf("\n\n%d %s %d\n\n",
20            (*std).id,std[0].name,(*std).age);
21
22     std++;
23     printf("\n\n%d %s %d\n\n",
24            (*std).id,(*std).name,(*std).age);
25 }
26
27 struct student *Input()
28 {
29    struct student *students = malloc(3*sizeof(struct student));
30    students[0].id = 1;
31    strcpy(students[0].name, "Jane");
32    (*students).age = 18;
33
34    students[1].id =2;
35    strcpy(students[1].name, "John");
36    (*(students+1)).age = 19;
37
38    return students;
39 }
```

```
1 Jane 18

2 John 19
```

Exercise 1

หาผลลัพธ์ที่ได้จากส่วนของโปรแกรม

```
#include <stdio.h>
typedef struct {
  int day,month,year;
} date;
void edit(date a);
int main() {
  date d1 = {15,10,1992};
  edit(d1);
  printf("%d/%d/%d\n",d1.day,d1.month,d1.year);
  return 0;
}
void edit(date a)
{
  a.year = a.year + 10;
}
```

Exercise 2

- ให้ ss เป็น Structure สำหรับเก็บข้อมูลตัวเลข 3 ตัว จงเขียนโปรแกรมภาษาซีเพื่อ
  - รับตัวเลขตัวที่ 1 และ 2 จากแป้นพิมพ์
  - สร้างฟังก์ชันสำหรับหาผลรวมของตัวเลขที่ 1 และ 2 ลงในตัวเลขที่ 3
  - ใช้การอ้างอิงข้อมูลโดย pointer เท่านั้น ไม่มีการส่งค่ากลับจากฟังก์ชัน

Exercise 3

```
1  #include <stdio.h>
2  typedef struct {
3      char id[12];
4      char name[50];
5      int age;
6      float grade;
7  } Student;
8
9  Student input2(Student *std);
10 void printCard(Student std);
11
12 void main()
13 {
14     Student newStd;
15     input2(&newStd);
16     printCard(newStd);
17 }
18 Student input2(Student *std)
19 {
20     scanf("%s", std->id);
21     scanf("%s", std->name);
22     scanf("%d", &std->age);
23     scanf("%f", &std->grade);
24 }
25 void printCard(Student std)
26 {
27    printf("Card Info.: %s %s %d years old\n", std.id,std.name,std.age);
28 }
```

อธิบายแต่ละบรรทัดดังต่อไปนี้

Line 2-7:

Line 9:

Line 10:

Line 14:

Line 15:

Line 16:

Line 18-24:

Line 25-28:

Recursion

- A process by which a function calls itself repeatedly
- The process is used for repetitive computations in which each action is stated in terms of a previous result

- **Two conditions** to solve a problem recursively
  - Write a problem in a recursive form
    - At lest one recursive case to call itself
    - Returning the result to its call
  - Must include at least one stopping condition

- Simple example of recursion

```
1  void recurse()
2  {
3      recurse();
4  }
5  void main()
6  {
7      recurse();
8  }
```

Example: Calculating factorials

- Suppose we wish to calculate the factorials of a positive integer
- Factorial
  - n! = 1 x 2 x 3 x … x n , n is a positive number
- Recursive form
  - n! = n x (n-1)!
- Stopping condition
  - 1! = 1 by definition

Example: Calculating factorials

- Problem : finding factorial of the number 4 using recursion technique
- 4! -> 1*2*3*4 is the set of the problem initially
- **Step 1** :
  - 4! = 4 x 3! -> 1*2*3 is the set of the problem now
- **Step 2**:
  - 4 x 3! = 4 x 3 x 2! -> 1*2 is the set of the problem now
- **Step 3**:
  - 4 x 3 x 2! = 4 x 3 x 2 x 1! -> 1! Is equal to 1. It is the stopping condition

Example: Calculating factorials

$4! = 4 \times 3!$

$3! = 3 \times 2!$

$2! = 2 \times 1!$

$1! = 1$

$n! = n \times (n-1)!$

$(n-1)! = (n-1) \times (n-2)!$

$(n-2)! = (n-1) \times (n-3)!$

.....

$2! = 2 \times 1!$

$1! = 1$

Example: Calculating factorials

```c
#include<stdio.h>

int fac(int n)
{
    if(n==1)
        return 1;
    else
        return (n*fac(n-1));
};
void main()
{
    int n;

    printf("Enter n: ");
    scanf("%d",&n);

    printf("n! = %d\n\n", fac(n));
}
```

```
Enter n: 4
n! = 24
```

Example

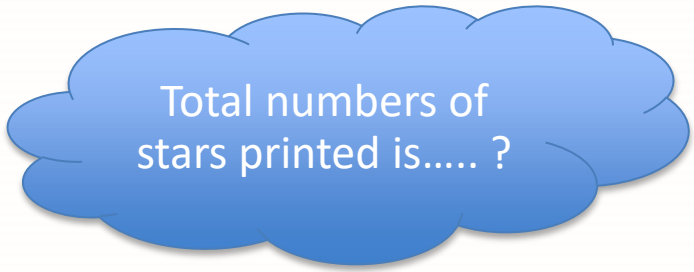
```
1   #include<stdio.h>
2
3   int f(int n)
4   {
5       if(n==1)
6           return 1;
7       else
8           return (n+f(n-1));
9   };
10  void main()
11  {
12      int n;
13
14      printf("Enter n: ");
15      scanf("%d",&n);
16
17      printf("%d\n\n", f(n));
18  }
```

Example

```
1  #include<stdio.h>
2  int f(int x, int y)
3  {
4    if(x == 0)
5      return y;
6    else
7      return f(x - 1,   x + y);
8  }
9
10
11 void main()
12 {
13     int n, m;
14
15     printf("Enter n: ");
16     scanf("%d",&n);
17
18     printf("Enter m: ");
19     scanf("%d",&m);
20
21     printf("%d\n\n", f(n,m));
22 }
```

Example

```c
#include<stdio.h>

void f(int n)
{
    int i = 0;
    if (n > 1)
      f(n-1);
    for (i = 0; i < n; i++)
      printf(" * ");
}
void main()
{
     int n;

     printf("Enter n: ");
     scanf("%d",&n);

     f(n);
}
```



- Advantage
  - Reducing the length of code
  - Easy to read code
  - Dealing with tree structures

- Disadvantage
  - Taking a lot of memory by copying memory for recursion depth
  - Causing stack overflow if it goes too deep to solve a problem

Exercise

- เขียนโปรแกรมภาษาซี เพื่อหาผลรวมของข้อมูลที่อยู่ใน array
- int A[5] = { 1, 2, 3, 4, 5 };
- พร้อมทั้งเขียนอธิบายการทำงานโปรแกรมที่เขียนพอเข้าใจ