FUNCTION

Natsuda Kaothanthong
CS102 (30 Sept. 2014)

LAST LECTURE

Control Structure:

- การหยุดวนรอบโดยใช้ break, exit, และ return
- การใช้ continue
- For
- do while
- Nested Loop

THIS LECTURE

- ทบทวนNested Loop
- Function
  - ประเภทของฟังชั่น
  - การเรียกใช้งานฟังชั่น
    - Caller
    - Callee
  - การจัดวางฟังชั่นที่สร้างขึ้น
  - Variable
    - Local Variable
    - Global Variable

ทบทวน NESTED LOOP

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int i, j, k;
6
7      for(i = 1; i < 6; i++)
8      {
9          for(j = 1; j < i; j++)
10         {
11             printf("*");
12         }
13         for(k = i; k < 6; k++)
14         {
15             printf("%d",k);
16         }
17         printf("\n");
18     }
19     return 0;
20 }
```

FUNCTION

ในภาษาซี ฟังชั่นแบบออกเป็น 2 ประเภทคือ

1. ฟังชั่นมาตรฐานของภาษาซี เช่น printf() และ scanf
2. ฟังชั่นที่โปรแกรมเมอร์สร้างขึ้นเอง

```
int main()
{
    คำสั่งภายในฟังชั่น main
    return 0;
}
```


ส่วนประกอบของฟังชั่นย่อยที่สร้างขึ้น

```
ชนิดฟังชั่น ชื่อฟังชั่น(ชนิดของข้อมูลที่ได้รับ ชื่อข้อมูล)
{
    คำสั่งภายใจฟังชั่น

    return ค่าส่งกลับ;
}
```


ประเภทของฟังชั่น 1

```
void myFunc1(void)
{
    คำสั่งภายใจฟังชั่น

}
```


ประเภทของฟังชั่น 2

```
void myFunc2(int x)
{
    คำสั่งภายใจฟังชั่น

}
```

ประเภทของฟังชั่น 3

```
double  myFunc3(void)
{
        double j;
        คำสั่งอื่น ๆ

        return j;
}
```


ประเภทของฟังชั่น 4

```
float  myFunc4(int k, float y, double d, char c)
{
        float result;
        คำสั่งภายใจฟังชั่น

        return  result;
}
```


การเรียกใช้ฟังชั่นย่อยที่สร้างขึ้น

การเรียกใช้ฟังชั่นประกอบด้วย

1. Caller: ฟังชั่นที่เรียกใช้อีกฟังชั่น
   - ข้อมูลที่ส่งไปเรียกว่า Parameter
2. Callee: ฟังชั่นที่ถูกเรียกใช้
   - ข้อมูลที่ได้รับเรียกว่า Argument

การเรียกใช้ฟังชั่นย่อยที่สร้างขึ้น(ต่อ)

```
int myFunc(int input)
{
    int num;
    . . . . . .
    return num;
}
```

```
int main()
{
    . . . . . .
    result = myFunc(int val1);
    . . . . . .
    return 0;
}
```

ตัวอย่างฟังชั่นประเภทที่ 1

```
void displayText( )
{
    printf("In displayText");
}
```

```
int main()
{
    displayText();
    return 0;
}
```


ตัวอย่างฟังชั่นประเภทที่ 2

```
void displaySum(int num)
{
    int result = num+5;
    printf("result is %d", result);
}
```

```
int main()
{
    int value = 3;
    displaySum(value);
    return 0;
}
```


ตัวอย่างฟังชั่นประเภทที่ 3

```
int  getValue()
{
    int x = 8;
    return x;
}
```

```
int main()
{
    int a;
    a = getValue();
    return 0;
}
```


ตัวอย่างฟังชั่นประเภทที่ 4

```
int  computeSum(int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}
```

```
int main()
{
    int x = 2, y = 3, result;
    result = computeSum(x,y);
    printf("Result is %d",result);
    return 0;
}
```

การจัดวางฟังชั่นที่สร้างขึ้น

1. วางฟังชั่นที่สร้างขึ้นไว้ก่อนถึง int main()
2. ใช้ฟังชั่น prototype เพื่อบอกโปรแกรมว่ามีฟังชั่นที่สร้างขึ้น

```
int  computeSum(int a, int b);

int main()
{
      int x = 2, y = 3, result;
      result = computeSum(x,y);
      printf("Result is %d",result);
      return 0;
}
int  computeSum(int a, int b)
{
      int sum;
      sum = a + b;
      return sum;
}
```


VARIABLE – LOCAL VARIABLE

```
int  computeSum(int a, int b)
{
      int sum;
      sum = a + b;
      return sum;
}
int main()
{
      int x = 2, y = 3, result;
      result = computeSum(x,y);
      printf("Result is %d",result);
      return 0;
}
```


VARIABLE – GLOBAL VARIABLE

```
int g;
int  computeSum(int a, int b)
{
      sum = a + b;
      g = a-b;
      return sum;
}
int main()
{
      int x = 2, y = 3, result;
      result = computeSum(x,y);
      printf("Result is %d",result);
      printf("g is %d",g);
      return 0;
}
```