ภาษา C กับ Microcontroller

ไพโรจน์ ครองตน

แผนกวิชาเครื่องมือวัดและควบคุม

รูปแบบโครงสร้างภาษา C

```
#include    …
```

การประกาศ Header file

```
#define   …
Char  …
int   …
```

การประกาศค่าคงที่และตัวแปรชนิดโกลบอล

```
Void     func1()
{

}
```

การสร้างฟังก์ชันหรือโปรแกรมย่อย

```
main()
{

}
```

การสร้างฟังก์ชันหลัก

ตัวแปรและชนิดของตัวแปร

ตัวแปร หมายถึงชื่อใช้แทนตำแหน่งของหน่วยความจำ RAM เก็บข้อมูลชั่วคราว

ชนิดตัวแปร	ขนาดข้อมูล	ค่าที่สามารถเก็บได้
char	1 byte	-128 ถึง 127
unsigned char	1 byte	0 ถึง 255
int	2 byte	-32,768 ถึง 32,767
long	2 byte	0 ถึง 65,535
unsigned long	4 byte	-2,147,483,648 ถึง 2,147,483,647
float	4 byte	$3.4X10^{-38}$ ถึง $3.4X10^{38}$
double	4 byte	$3.4X10^{-38}$ ถึง $3.4X10^{38}$

กฎการตั้งชื่อตัวแปร

1. ขึ้นต้นด้วยตัวอักษรและไม่มีอักษรพิเศษ เช่น ! @ # $ % ^ & | * ( )
2. ใช้เครื่องหมาย Underscore (_) แยกคำตัวแปร
3. ห้ามมีช่องว่าง
4. มีตัวเลขได้แต่ห้ามขึ้นต้นชื่อด้วยตัวเลข
5. ใช้อักษรได้ทั้งตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่ (case sensitive)
6. ห้ามตั้งชื่อตัวแปรซ้ำกับคำสงวนหรือคำสั่งในภาษา C

วิธีสร้างตัวแปรและการกำหนดค่า

ตัวแปรอาร์เรย์

ตัวแปรอาร์เรย์(Array) คือตัวแปรที่สามารถเก็บข้อมูลได้เป็นชุด
โดยข้อมูลที่เก็บต้องเป็นชนิดเดียวกัน

ชนิดของข้อมูล ชื่อของตัวแปรอาร์เรย์ [จำนวนสมาชิกที่กำหนดให้มี]

```
char A [5]
```

ประกาศตัวแปรอาร์เรย์ A เพื่อเก็บข้อมูล
ชนิด char จำนวน 5 สมาชิก

```
unsigned char segment[ ] = {0x01,0x4F,0x12,0x06,0x4c};
```

การกำหนดให้ตัวแปร A
เท่ากับ 0x12 ในตัวแปร segment

```
A = segment [2];
```

ค่าคงที่ในภาษา C

ค่าข้อมูลที่กำหนดในให้เป็นค่าคงที่ไม่สามารถเปลี่ยนแปลงหรือแก้ไขได้ตลอดการทำงานของโปรแกรม

1) คำสั่ง #define **2) คำสั่ง const**

```
#define MAX 100
#define MIN 0
const float PI = 3.14;
void main()
{
        float Area,R;
        R = 2.0;
        Area = PI*R*R
}
```

เครื่องหมายดำเนินการในภาษา C

เครื่องหมายดำเนินการกำหนดค่า (Assignment Operator) กำหนดค่าให้กับตัวแปร เช่น a = 10

เครื่องหมายการคำนวนทางคณิตศาสตร์(Arithmetric Operator)

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	การบวก	C = A + B
-	การลบ	C = A - B
*	การคูณ	C = A * B
/	การหาร	C = A / B
%	การหาเศษจากการหาร	C = 7 % 5 , เท่ากับ 2
- -	การลดค่าลงหนึ่ง	A - - มีค่าเท่ากับ A = A - 1
+ +	การเพิ่มค่าขึ้นหนึ่ง	A + + มีค่าเท่ากับ A = A + 1

เครื่องหมายดำเนินการในภาษา C

เครื่องหมายการดำเนินการระดับบิต(Bitwise Operator)

เครื่องหมาย	ความหมาย	ตัวอย่าง
~	NOT หรือ COMPLETEMENT	C = ~ A
&	การAND	C = A & B
|	การOR	C = A | B
^	การXOR	C = A ^ B
<<	การเลื่อนข้อมูลไปทางซ้าย	C = A << 1
>>	การเลื่อนข้อมูลไปทางขวา	C = A >> 1

เครื่องหมายดำเนินการในภาษา C

เครื่องหมายการเปรียบเทียบ(Relational Operator)

เครื่องหมาย	ความหมาย	ตัวอย่าง
>	มากกว่า	A > B (A มากกว่า B)
>=	มากกว่าหรือเท่ากับ	A >= B (A มากกว่าหรือเท่ากับ B)
<	น้อยกว่า	A < B (A น้อยกว่า B)
<=	น้อยกว่าหรือเท่ากับ	A <= B (A น้อยกว่าหรือเท่ากับ B)
= =	เท่ากับ	A = = B (A เท่ากับ B)
!=	ไม่เท่ากับ	A != B (A ไม่เท่ากับ B)

เครื่องหมายดำเนินการในภาษา C

เครื่องหมายดำเนินการทางลอจิก (Logical Operator)

เครื่องหมาย	ความหมาย	ตัวอย่าง
&&	เอาสถานะของเงื่อนไขมา AND กัน	C = (1 < 2) && (3 > 4) ได้ผลลัพธ์เป็นเท็จ
||	เอาสถานะของเงื่อนไขมา OR กัน	C = (1 < 2) || (3 > 4) ได้ผลลัพธ์เป็นจริง
!=	เอาสถานะของเงื่อนไขทางขวามือให้มีสถานะตรงกันข้าม	C = !(1 < 2) ได้ผลลัพธ์เป็นเท็จ

คำสั่งตรวจสอบเงื่อนไข

คำสั่ง if : สำหรับการตรวจสอบเงื่อนไขชนิดทางเลือกเดียว

```
If (เงื่อนไขการเปรียบเทียบ)
{
        ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบเป็นจริง
}
```

```
void main()
{
        char A,B,Max;
        A = 20;
        B = 10;
        if (A > B)
    {
        Max = A;
     }
}
```

คำสั่งตรวจสอบเงื่อนไข

คำสั่ง if - else : สำหรับการตรวจสอบเงื่อนไขชนิด 2 ทางเลือก

```
if (เงื่อนไขการเปรียบเทียบ)
{
    ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบเป็นจริง
}
else
{
    ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบเป็นเท็จ
}
```

คำสั่งตรวจสอบเงื่อนไข

คำสั่ง if - else : สำหรับการตรวจสอบเงื่อนไขชนิด 2 ทางเลือก

```
void main()
    {
        char A,B,Max;
        A = 20;
        B = 10;
        if (A>B)
        {
            Max = A;
        }
            else
        {
        Max = B;
        }
    }
```

คำสั่ง if – else - if : สำหรับการตรวจสอบเงื่อนไขชนิดหลายทางเลือก

```
if (เงื่อนไขการเปรียบเทียบ#1)
{
    ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบ#1เป็นจริง
}
    else if (เงื่อนไขการเปรียบเทียบ#2)
{
    ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบ#2เป็นจริง
}
 else if (เงื่อนไขการเปรียบเทียบ#N)
{
    ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบ#Nเป็นจริง
}
else
{
    ชุดคำสั่งที่ต้องการให้ทำงานเมื่อเงื่อนไขการเปรียบเทียบ#Nเป็นเท็จ
}
```

คำสั่ง if – else - if : สำหรับการตรวจสอบเงื่อนไขชนิดหลายทางเลือก

```
void main()
{
    char Operator;
    int A,B,C;
    A = 100;
    B = 200;
    Operator = '+';
    if (Operator = = '+')
    {   C = A + B
    }
     else if (Operator = = '-')
          {     C = A-B;
          }
```

```
else if (Operator = = '*')
          {     C = A*B;
          }
           else if (Operator = = '/')
                {     C = A/B;
                }
                else
                     {     C = 0;
                     }
}
```

คำสั่ง switch - case

ตรวจสอบตัวแปรหนึ่งตัวกับหลายๆเงื่อนไข

```
switch (ตัวแปรที่ต้องการตรวจสอบ)
{
    case ค่าที่1 : ชุดคำสั่งที่ต้องการให้ทำงาน
    case ค่าที่2 : ชุดคำสั่งที่ต้องการให้ทำงาน
    ....
    default :  ชุดคำสั่งที่ต้องการให้ทำงานกรณีที่ไม่มีเงื่อนไขไม่เป็นจริงเลย
{
```

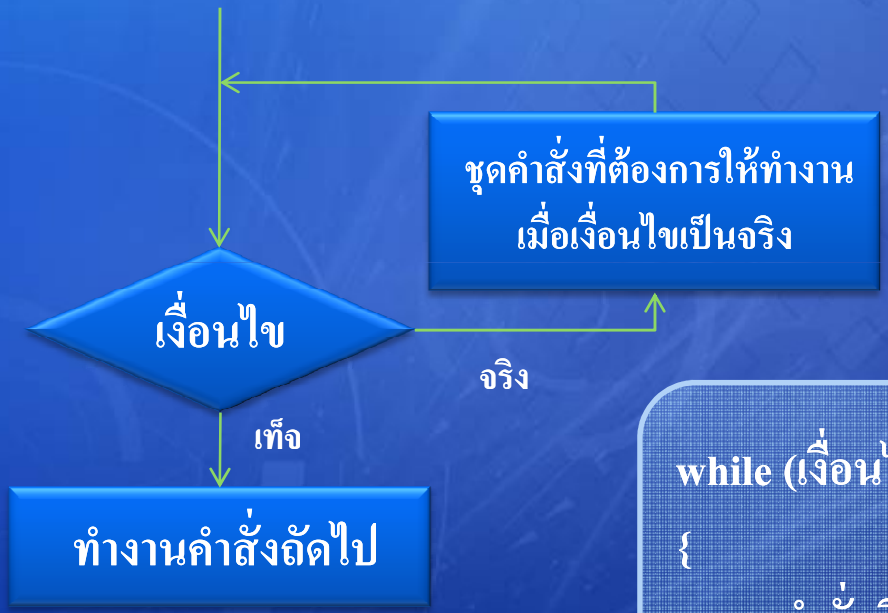
คำสั่ง switch - case

ตรวจสอบตัวแปรหนึ่งตัวกับหลายๆเงื่อนไข

```
void main()
{
        char Operator;
        int A,B,C;
        A = 100;
        B = 200;
        Operator = '+';
        switch (Operator)
        {
            case '+': C = A+B;          Break;
            case '-': C = A-B;          Break;
            case '*': C = A*B;          Break;
            case '/': C = A/B;          Break;
           default:  C = 0 ;
        }
    }
```

คำสั่ง การทำซ้ำหรือวนลูป

คำสั่ง while วนลูปตราบใดที่เงื่อนไขในการตรวจสอบมีค่าเป็นจริง




```
while (เงื่อนไข)
{
    ชุดคำสั่งที่ต้องการให้ทำงาน เมื่อเงื่อนไขเป็นจริง
{
```

คำสั่ง การทำซ้ำหรือวนลูป

คำสั่ง while วนลูปตราบใดที่เงื่อนไขในการตรวจสอบมีค่าเป็นจริง

```
void main()
{
        char A,B;
        A = 0;
        while (A<B);
        {
                A = A+1;
        }
        B = A+10;
}
```

คำสั่ง การทำซ้ำหรือวนลูป

คำสั่ง do - while วนลูปตราบใดที่เงื่อนไขในการตรวจสอบมีค่าเป็นจริง สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นเท็จ


```
do
{
    ชุดคำสั่งที่ต้องการให้ทำงาน เมื่อเงื่อนไขเป็นจริง
{
while (เงื่อนไข)
```

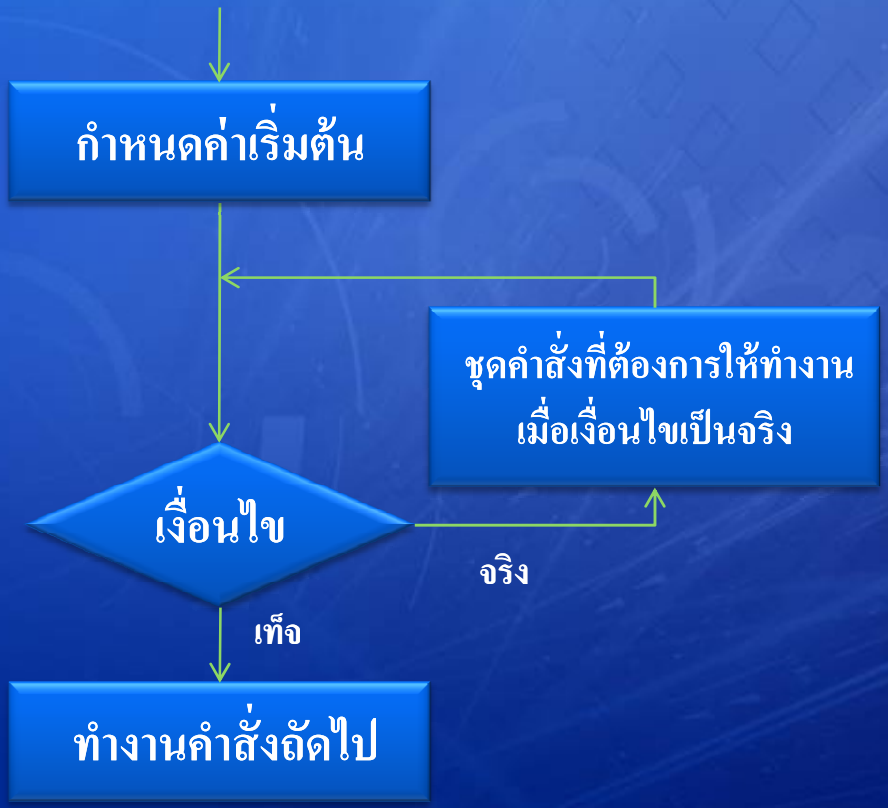
คำสั่ง การทำซ้ำหรือวนลูป

คำสั่ง do - while วนลูปตราบใดที่เงื่อนไขในการตรวจสอบมีค่าเป็นจริง

```
void main()
{
        char A,B;
        A = 0;
        do
        {
                A = A+1;
        }
        while (A<B);
        B = A+10;
}
```

คำสั่ง การทำซ้ำหรือวนลูป

คำสั่ง for วนลูปตราบใดที่เงื่อนไขในการตรวจสอบมีค่าเป็นจริง สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นเท็จ




```
for (กำหนดค่าเริ่มต้นตัวนับ;
      เงื่อนไขเปรียบเทียบ;เพิ่มลดค่าตัวนับ )
{
   ชุดคำสั่งที่ต้องการให้ทำงาน
{
```

คำสั่ง การทำซ้ำหรือวนลูป

คำสั่ง for วนลูปตราบใดที่เงื่อนไขในการตรวจสอบมีค่าเป็นจริง สิ้นสุดการทำงานเมื่อเงื่อนไขเป็นเท็จ

```
void main()
{
        char a;
        int Total = 0;
         for (a = 1;a<=100;a++)
        {
              Total = Total+a;
         }
    }
```

การสร้างและใช้งานฟังก์ชัน

ฟังก์ชันที่มาพร้อมกับคอมไพลเลอร์(compiler function)

#include<stdlib.h> กลุ่มคำสั่งการแปลงค่าข้อมูล เช่น abs , atof , atoi , atoll

#include<string.h> กลุ่มคำสั่งการจัดการข้อมูลประเภทข้อความเช่น strcpy , strlen , strcmp

การสร้างและใช้งานฟังก์ชัน

ฟังก์ชันที่ต้องเขียนขึ้นเอง(user defined function)

```
#include <ชื่อไลบารี่>
void func1()
{
  ชุดคำสั่งการทำงานของฟังก์ชัน
}
void main()
{
 ....
 ....
Function1();      //การเรียกใช้ฟังก์ชัน func1
}
```

การสร้างและใช้งานฟังก์ชัน

```
#include <ชื่อไลบารี่>
void func1()
void func2()
void main()
{
 ....
Function1();      //การเรียกใช้ฟังก์ชัน func1
Function2();      //การเรียกใช้ฟังก์ชัน func2
}
void func1()
{
ชุดคำสั่งการทำงานของฟังก์ชัน func1
}
void func2()
{
ชุดคำสั่งการทำงานของฟังก์ชัน func2
}
```

รูปแบบการประกาศใช้งานฟังก์ชัน

```
#include <ชื่อไลบารี่>
void func1();
void func2();
void main()
{
 ....
Function1();      //การเรียกใช้ฟังก์ชัน func1
Function2();      //การเรียกใช้ฟังก์ชัน func2
}
void func1()
{
ชุดคำสั่งการทำงานของฟังก์ชัน func1
}
void func2()
{
ชุดคำสั่งการทำงานของฟังก์ชัน func2
}
```

การสร้างฟังก์ชันแบบธรรมดา

```
#include <m8c.h>
void delay();
void main()
{
   while(1)
    {
       PRT1DR = 0x01;     //ส่งข้อมูลออกไปยังพอร์ต PRT1DR
       Delay();             //เรียกฟังก์ชันหน่วงเวลา
       PRT1DR = 0x00;     //ส่งข้อมูลออกไปยังพอร์ต PRT1DR
       Delay();             //เรียกฟังก์ชันหน่วงเวลา
    }
}
void delay()
{
      unsigned int loop;
    for (Loop = 0;Loop<=25;Loop++);
}
```

การสร้างฟังก์ชันแบบรับค่าพารามิเตอร์เข้า

```
#include <m8c.h>
void delay(unsigned int Loop);
void main()
{
   while(1)
    {
       PRT1DR = 0x01;     //ส่งข้อมูลออกไปยังพอร์ต PRT1DR
       Delay(25000);      //หน่วงเวลา 25,000 รอบ
       PRT1DR = 0x00;     //ส่งข้อมูลออกไปยังพอร์ต PRT1DR
       Delay(25000);      //หน่วงเวลา 25,000 รอบ
    }
}
void delay(unsigned int Loop)
{
      unsigned int i;
    for (i = 0;i<=Loop;i++);
}
```

การสร้างฟังก์ชันแบบรับค่าพารามิเตอร์ออก

```
int Converter_to_F(int C_Unit);
void main()
{
   int C,F;
   C = 25;
    F = Convert_to_F(C);
   }
    int Convert_to_F(int C_Unit)
   {
    return ((9/5)*C_Unit+32);
   }
```

ตัวแปรภายใน(Local Variable)

```
void main()
{
   char a,b;   //ตัวแปรที่สามารถใช้ได้เฉพาะภายในฟังก์ชัน main
   …
}
    void func1()
    {
     char a;  //ตัวแปรที่สามารถใช้ได้เฉพาะภายในฟังก์ชัน func1
     …
    }
```

ตัวแปรภายนอก(Global Variable)

```
Char a;
void main()
{
   char a,b;   //ตัวแปรที่สามารถใช้ได้เฉพาะภายในฟังก์ชัน main
   …
}
    void func1()
    {
     void func1()
     {
      a = 20;
      …
     }
```

Thank You !

Krongton.tatc.ac.th