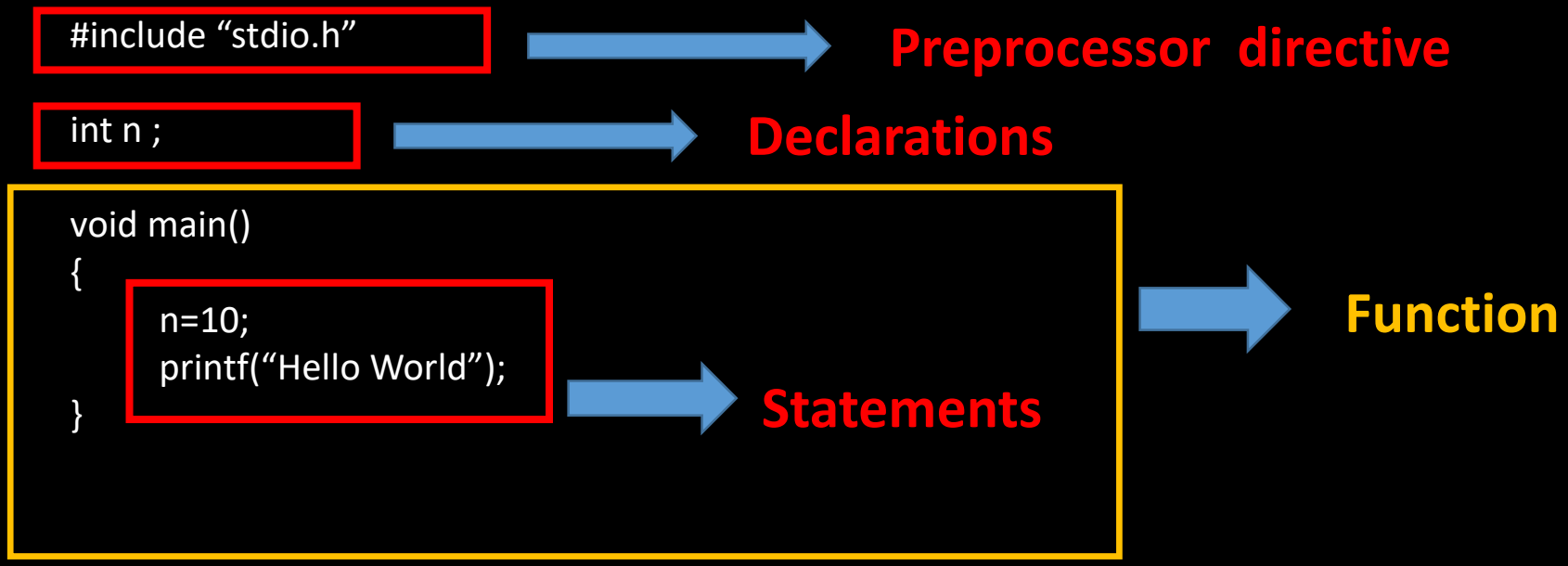
BASIC C++ PROGRAMMING

- Basic C++ Structor
- Variable & Input
- If, If-else, Switch case
- Loop
- Operator

ตัวแปร Variable



คำสั่งแสดงผลบนหน้าจอ

printf(“ข้อมูลที่ต้องการแสดง”,ตัวแปร);

EX. **printf(“Hello World”);**

printf(“%d”,10);

รหัส	ความหมาย
% c	ใช้กับตัวแปรที่เก็บค่าที่เป็นตัวอักษรเพียงตัวเดียว
% s	ใช้กับตัวแปรที่เก็บค่าที่เป็นข้อความที่เก็บในตัวแปรชุด
% d	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขจำนวนเต็ม
% u	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขจำนวนเต็มบวก
% f	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขทศนิยม
% e	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขทศนิยมในรูป e ยกกำลัง
% x	ใช้กับตัวแปรที่เก็บค่าที่เป็นค่าเลขฐานสิบหก
% o	ใช้กับตัวแปรที่เก็บค่าที่เป็นค่าเลขฐานแปด
% p	ใช้กับตัวแปรที่เก็บค่าที่เป็นตัวชี้ตำแหน่ง (pointer)

คำสั่งรับค่าจาก Keyboard

scanf(“ข้อมูลที่ต้องการแสดง”,ตัวแปร);

EX. int a;

scanf(“%d”,&a);

รหัส	ความหมาย
% c	ใช้กับตัวแปรที่เก็บค่าที่เป็นตัวอักษรเพียงตัวเดียว
% s	ใช้กับตัวแปรที่เก็บค่าที่เป็นข้อความที่เก็บในตัวแปรชุด
% d	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขจำนวนเต็ม
% u	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขจำนวนเต็มบวก
% f	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขทศนิยม
% e	ใช้กับตัวแปรที่เก็บค่าที่เป็นเลขทศนิยมในรูป e ยกกำลัง
% x	ใช้กับตัวแปรที่เก็บค่าที่เป็นค่าเลขฐานสิบหก
% o	ใช้กับตัวแปรที่เก็บค่าที่เป็นค่าเลขฐานแปด
% p	ใช้กับตัวแปรที่เก็บค่าที่เป็นตัวชี้ตำแหน่ง (pointer)

ตัวแปร Variable

Type Variable_Name,Variable_Name ...;

Ex. `int n1;` ประกาศให้ตัวแปร n1 คือตัวแปรที่เก็บข้อมูลชนิดเลขจำนวนเต็ม

ชนิดของตัวแปร	ขนาด (bits)	ขอบเขต	ข้อมูลที่เก็บ
char	8	-128 ถึง 127	ข้อมูลชนิดอักขระ ใช้เนื้อที่ 1 byte
unsigned char	8	0 ถึง 255	ข้อมูลชนิดอักขระ ไม่คิดเครื่องหมาย
int	16	-32,768 ถึง 32,767	ข้อมูลชนิดจำนวนเต็ม ใช้เนื้อที่ 2 byte
unsigned int	16	0 ถึง 65,535	ข้อมูลชนิดจำนวนเต็ม ไม่คิดเครื่องหมาย
short	8	-128 ถึง 127	ข้อมูลชนิดจำนวนเต็มแบบสั้น ใช้เนื้อที่ 1 byte
unsigned short	8	0 ถึง 255	ข้อมูลชนิดจำนวนเต็มแบบสั้น ไม่คิดเครื่องหมาย
long	32	-2,147,483,648 ถึง 2,147,483,649	ข้อมูลชนิดจำนวนเต็มแบบยาว ใช้เนื้อที่ 4 byte
unsigned long	32	0 ถึง 4,294,967,296	ข้อมูลชนิดจำนวนเต็มแบบยาว ไม่คิดเครื่องหมาย
float	32	3.4*10e(-38) ถึง 3.4*10e(38)	ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 4 byte
double	64	3.4*10e(-308) ถึง 3.4*10e(308)	ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 8 byte
long double	128	3.4*10e(-4032) ถึง 1.1*10e(4032)	ข้อมูลชนิดเลขทศนิยม ใช้เนื้อที่ 16 byte

ตัวแปร Variable

```
#include “stdio.h”

int number1;

void sum()
{
      int number2;
      number1 = 13;
      number2 = 12;

}

main()
{
      int number3;
      number1 = 13;
      number3 = 11;
}
```

Global

Local

Local

ค่าคงที่ Constants

การประกาศค่าคงที่ทำได้ 2 แบบ

- **const type variable_name = value;**

  const int runTime = 100;

- **#define constants_name value**

  #define runtime 100

ตัวดำเนินการ Operators

ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการ	กระบวนการ	ข้อมูลที่ถูกกระทำ	ข้อมูลผลลัพธ์
+	บวก (Addition)	จำนวนเต็ม,จำนวนจริง	จำนวนเต็ม,จำนวนจริง
-	ลบ (Subtraction)	จำนวนเต็ม,จำนวนจริง	จำนวนเต็ม,จำนวนจริง
*	คูณ (Multiplication)	จำนวนเต็ม,จำนวนจริง	จำนวนเต็ม,จำนวนจริง
/	หาร (Real Number Division)	จำนวนเต็ม,จำนวนจริง	จำนวนจริง
%	การหารแบบเอาเศษ (Modulus)	จำนวนเต็ม	จำนวนเต็ม

ตัวดำเนินการ Operators

ตัวดำเนินการทางเปรียบเทียบ

ตัวดำเนินการ	การกระทำ
==	เท่ากับ
!=	ไม่เท่ากับ
<=	น้อยกว่าหรือเท่ากับ
>=	มากกว่าหรือเท่ากับ
>	มากกว่า
<	น้อยกว่า

ตัวดำเนินการ Operators

ตัวดำเนินการทางตรรกะ

ตัวดำเนินการ	การกระทำ
&&	ดำเนินการ AND ค่าสองค่า ถ้าค่าทั้งสองเป็นจริง ผลลัพธ์จะเป็นจริง
||	ดำเนินการ OR ค่าสองค่า ถ้าค่าทั้งสองเป็นเท็จ ผลลัพธ์จะเป็นเท็จ
!	ดำเนินการ NOT เปลี่ยนค่า จากจริงเป็นเท็จ จากเท็จเป็นจริง

ตัวดำเนินการ Operators

ตัวดำเนินการเพิ่มค่าลดค่า

ตัวดำเนินการ	นิพจน์	ความหมาย
++ (Prefix)	++a	เพิ่มค่าให้กับ a หนึ่งค่าก่อน จึงนำค่าใหม่ของ a ในนิพจน์นี้ไปใช้
++ (Postfix)	a++	นำค่าปัจจุบันของ a ในนิพจน์นี้ไปใช้ก่อน จึงเพิ่มค่าให้กับ a หนึ่งค่า
-- (Prefix)	--b	ลดค่าให้กับ b หนึ่งค่าก่อน จึงนำค่าใหม่ของ b ในนิพจน์นี้ไปใช้
-- (Postfix)	b--	นำค่าปัจจุบันของ b ในนิพจน์นี้ไปใช้ก่อน จึงลดค่าให้กับ b หนึ่งค่า

การตรวจสอบเงื่อนไข

รูปแบบการตรวจสอบเงื่อนไข if

```
if (เงื่อนไข)
{
        คำสั่ง;
}
```

- ถ้าเงื่อนไขเป็นจริง จะทำคำสั่งภายบล็อกปีกกาของ if
- ถ้าเงื่อนไขเป็นเท็จ จะทำคำสั่งต่อจากบล็อกปีกาปิดของ if

การตรวจสอบเงื่อนไข

รูปแบบการตรวจสอบเงื่อนไข if ... else

```
if (เงื่อนไข)
{
        คำสั่ง;
}
else
{
        คำสั่ง
}
```

- ถ้าเงื่อนไขเป็นจริง จะทำคำสั่งภายในบล็อกปีกกาของ if
- ถ้าเงื่อนไขเป็นเท็จ จะทำคำสั่งภายในบล็อกปีกกาของ else

การตรวจสอบเงื่อนไข

รูปแบบการตรวจสอบเงื่อนไข if ... else if ....

```
if (เงื่อนไข 1)
{
        คำสั่ง;
}
else if (เงื่อนไข 2)
{
        คำสั่ง
}
```

- ถ้าเงื่อนไข 1 เป็นจริง จะทำคำสั่งภายในบล็อกปีกกาของ if
- ถ้าเงื่อนไขเป็นเท็จ จะตรวจสอบเงื่อนไข 2 ถ้าเป็นจริงถึงจะทำ

การตรวจสอบเงื่อนไข

รูปแบบการตรวจสอบเงื่อนไข switch case

```
switch (ตัวแปรที่จะตรวจสอบ) {
      case ค่าที่สั่งให้ทำงาน:
         สิ่งที่ต้องการให้ทำ;
         break;
    default:
       สิ่งที่ต้องการให้ทำ
          break;
}
```

ลูป Loop

รูปแบบลูป For

```
for (ค่าเริ่มต้น; เงื่อนไข ;ส่วนเปลี่ยนแปลงค่า)
{
        คำสั่ง;
}
```

-ค่าเริ่มต้น เป็นส่วนกำหนดค่าเริ่มต้นการทำงานของลูป for

-เงื่อนไข เป็นส่วนกำหนดเงื่อนไขในการวนทำคำสั่งในลูป for

-ส่วนเปลี่ยนแปลงค่า เป็นส่วนการเปลี่ยนแปลงค่าในการวนทำคำสั่งในลูป for

ลูป Loop

ตัวอย่างคำสั่งลูป for

```
int result=0;
for(int i=1 ; i<=8; i++)
{
        result += i;
}
```

จากโค๊ดตัวอย่างค่าเริ่มต้นของตัวนับ i = 1 และทุกครั้งที่วนทำคำสั่งในบล็อกของ for ก็จะเพิ่มค่าตัวนับขึ้นทีละ 1 ค่า และทุกครั้งก่อนทำคำสั่งในบล็อก เงื่อนไข i<=8 จะต้องเป็นจริงทุกครั้ง ถ้าเป็นเท็จโปรแกรมหลุดออกจากลูป for ทันที นั่นคือวนทำงานที่ i=1,2,3,4,5,6,7 และ 8 ในแต่ละรอบจะนำค่า i ไปบวกเพิ่มสะสมในตัวแปร result ซึ่งก็คือ

result =1+2+3+4+5+6+7+8

reslut = 36

ลูป Loop

ตัวอย่างการใช้ break ออกจากลูป for

```
int result=0;
for(int i=1 ; i<=8; i++)
{
    result += i;
    if (i==4)
    {
        break;
    }
}
```

จากโค๊ดตัวอย่างค่าเริ่มต้นของตัวนับ i = 1 และทุกครั้งที่วนทำคำสั่งในบล็อกของ for ก็จะเพิ่มค่าตัวนับขึ้นทีละ 1 ค่า และทุกครั้งก่อนทำคำสั่งในบล็อก เงื่อนไข i<=8 จะต้องเป็นจริงทุกครั้ง ถ้าเป็นเท็จโปรแกรมหลุดออกจากลูป for ทันที นั่นคือวนทำงานที่ i=1,2,3 และ 4 ในแต่ละรอบจะนำค่า i ไปบวกเพิ่มสะสมในตัวแปร result เมื่อเจอเงื่อนไข if(i==4) เป็นจริงจะหลุดออกจากลูป for ด้วยคำสั่ง break

result =1+2+3+4

reslut = 10

ลูป Loop

รูปแบบลูป while

```
while(เงื่อนไข)
{
        คำสั่ง;
}
```

-เงื่อนไข เป็นส่วนกำหนดเงื่อนไขในการวนทำคำสั่งภายในลูป while โดยถ้าเงื่อนไขเป็นจริง จะทำงานภายในบล็อกคำสั่ง while และวนกลับไปตรวจสอบเงื่อนไขอีกครั้ง ถ้าเมื่อไหร่ก็ตามที่เงื่อนไขเป็นเท็จโปรแกรมจะหลุดออกจากลูป while ทันที

ลูป Loop

ตัวอย่างคำสั่งลูป while

```
int count=5;
int result=0;
while(count)
{
        result += count;
        count--;
}
```

จากโค๊ดตัวอย่างตัวแปร count เริ่มต้นที่ 5 และทุกครั้งที่วนทำคำสั่งในบล็อกของ while ก็จะลดค่าลงครั้งละ 1 ค่า หลังจากการบวกค่าสะสมของตัวแปร count จนกระทั่งค่าตัวแปร count ถูกลดค่าลงจนเป็น 0 จะทำให้ตำแหน่งเงื่อนไข while เป็น 0 ตามไปด้วย ทำให้โปรแกรมหลุดออกจากลูป while ทันที นั่นคือวนทำงานที่ count=5,4,3,2,และ 1 ในแต่ละรอบจะนำค่า count ไปบวกเพิ่มสะสมในตัวแปร result ซึ่งก็คือ

result =5+4+3+2+1

reslut = 15

ลูป Loop

ตัวอย่างการใช้ break ออกจากลูป while

```
int count=5;
int result=0;
while(count)
{
        result += count;
        if(count==3)
        {
            break;
        }
        count--;
}
```

จากโค๊ดตัวอย่างตัวแปร count เริ่มต้นที่ 5 และทุกครั้งที่วนทำคำสั่งในบล็อกของ while ก็จะลดค่าลงครั้งละ 1 ค่า หลังจากการบวกค่าสะสมของตัวแปร count จนกระทั่งค่าตัวแปร count ถูกลดค่าลงจนเป็น 3 จะทำให้ตำแหน่งเงื่อนไข if(count==3) เป็นจริง ทำให้โปรแกรมหลุดออกจากลูป while ด้วยคำสั่ง break นั่นคือวนทำงานที่ count=5,4 และ 3 เท่านั้น ในแต่ละรอบจะนำค่า count ไปบวกเพิ่มสะสมในตัวแปร result ซึ่งก็คือ

result =5+4+3 reslut = 12

การสร้าง FUNCTION

รูปแบบ function แบบไม่มีการรับค่าและคืนค่า

```
void function_name()
{
        คำสั่ง;
}
```

-void คือคีย์เวิร์ดส่วนที่แจ้งว่าฟังก์ชั่นนี้ไม่มีการคืนค่าจากการประมวลผลใดๆออกมาจากฟังก์ชั่น

-function_name คือชื่อฟังก์ชั่นซึ่งจะต้องให้ตรงกับหลักการตั้งชื่อฟังก์ชั่น

ภายในบล็อกปีกกา{} ของฟังก์ชั่นให้ทำการบรรจุชุดคำสั่งต่างๆได้ตามต้องการ

การสร้าง FUNCTION

รูปแบบ function แบบมีการรับค่าแต่ไม่มีการคืนค่า

```
void function_name(var1_type var1,var2_type var2, var3_type var3,...)
{
        คำสั่ง;
        ……
}
```

-void คือคีย์เวิร์ดส่วนที่แจ้งว่าฟังก์ชั่นนี้ไม่มีการคืนค่าจากการประมวลผลใดๆออกมาจากฟังก์ชั่น

-var1_type, var2_type, var3_type,.. คือชนิดข้อมูลของตัวแปรอะกิวเมนต์ที่มารับค่าเข้าไปประมวลผลภายในฟังก์ชั่น

-var1, var2, var3,.. คือชื่อของตัวแปรอะกิวเมนต์ที่มารับค่าเข้าไปประมวลผลภายในฟังก์ชั่น

-ภายในบล็อกปีกกา{} ของฟังก์ชั่นให้ทำการบรรจุชุดคำสั่งต่างๆได้ตามต้องการ

การสร้าง FUNCTION

รูปแบบ function แบบมีการรับค่าและการคืนค่า

```
return_type function_name(var1_type var1,var2_type var2, var3_type var3,…)
{
        คำสั่ง;
        ……
        return value;
}
```

-return_type คือชนิดข้อมูลที่มีการคืนค่าออกมาจากฟังก์ชั่น

-var1_type, var2_type, var3_type,.. คือชนิดข้อมูลของตัวแปรอะกิวเมนต์ที่มารับค่าเข้าไปประมวลผลภายในฟังก์ชั่น

-var1, var2, var3,.. คือชื่อของตัวแปรอะกิวเมนต์ที่มารับค่าเข้าไปประมวลผลภายในฟังก์ชั่น

-ภายในบล็อกปีกกา{} ของฟังก์ชั่นให้ทำการบรรจุชุดคำสั่งต่างๆได้ตามต้องการ

-return คือคำสั่งคืนค่าและออกจากฟังก์ชั่น

-value คือค่าของข้อมูลที่คืนออกมาจากฟังก์ชั่นซึ่งจะต้องสอดคล้องกับ return_type