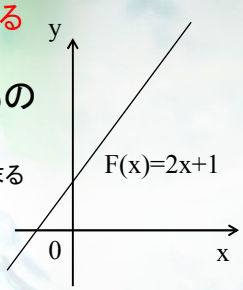
関数：Function

- 関数を使うと、処理を纏めて管理することができる
  → 処理の再実行（再利用）が簡単にできる。
- 関数とは入力値に対する値を定義するもの
  - 数学では、F(x) = 2x + 1　など
    ↑入力値(引数) x が決まれば、F(x)が決まる
- プログラムにおける関数とは：
  - 入力値を受取り、処理をして、値を返す（戻す）



- 関数の書式(戻り値が無い場合は　void型として指定する)

```
戻り値のデータ型　関数名（引数のデータ宣言の並び）
{
    ・・・
    return(戻り値);
}
```

例

```
int  func1(int x){
    x = 2*x + 1;
    return x ;
}
```

関数に引数として数値を渡す例

```
#include "stdafx.h"
int plus(int a, int b) { /* 足し算をする関数 */
    int c;
     c = a + b;
    return c;
}

int _tmain(int argc, _TCHAR* argv[ ])
{
    int a, b, c;
    a = 5;  b = 2;
    c = 5+2;
    printf(" 5 + 2 = %d¥n ", c); /*下の例でも結果は同じになる */
    printf(" 5 + 2 = %d¥n", plus(a, b));
    getchar();
}
```

関数の再利用例　　プロジェクト名：myFunc

```
#include "stdafx.h"
/* 使い回しできる処理は関数化して再利用しましょう*/
int plus(int a, int b) {  return a + b; }  /* 足し算をする関数*/
int minus(int a, int b) {  return a - b; }  /* 引き算をする関数*/

int _tmain(int argc, _TCHAR* argv[ ])
{
    printf("5 + 2 = %d¥n", plus(5, 2) );
    printf("3 - 1 = %d¥n", minus(3, 1) );
    printf("2 + 3 = %d¥n", plus(2, 3) );
    printf("2 - 5 = %d¥n", minus(2, 5) );
    getchar();
}
```

関数を定義しておけば、再利用できます。
この例では、plusやminusを上部(main関数の前)で定義しているので、main関数内でいつでも使う事が出来ます。

課題

- 2つの整数の四則演算を計算したい。
- 以下の関数を作って、計算させて下さい。

足し算をする関数	int plus(int a, int b) { }
引き算をする関数	int minus(int a, int b) { }
掛け算をする関数	int multi(int a, int b) { }
足し算をする関数	int divide(int a, int b) { }

プロジェクト名：myFunc
（このテキストの関数の再利用例を参考にして下さい）

関数に引数として配列を渡す例

プロジェクト名 : useFunc

```
#include "stdafx.h"
void dataSet (float y[ ], int n ) {
    int i;  for( i=0; i<5; i++ ){ y[i] = (float) i; }
}
void dataPrint (float x[ ], int n) {
    int i;  for( i=0; i<n; i++ ){  printf(" %f ¥n", x[i] ); }
}
int maxData (float y[ ], int n ) {
    float max=-999.0; int i;
    for( i=0; i<5; i++ ){ if(y[i] > max ) { max = y[i]; } }
    return max;
}
int _tmain(int argc, _TCHAR* argv[ ])
    int i;   float  a[5];
    dataSet( a, 5);
    dataPrint( a , 5); getchar();
}
```

dataSet関数は配列aにデータを代入します

dataPrint関数は配列aの内容を表示します

maxData関数は配列aの最大値を返します

変数のスコープ

- 変数には、その変数の有効な範囲（スコープ）がある
- ある関数内で宣言した変数はほかの関数から参照することはできない
- 関数の外に変数の宣言をおくと、どこの関数からも参照できるグローバル(全体)変数になる

変数の値をほかの関数で使いたいときは引数で渡す

（グローバル変数は一見大変便利に見えるが、長いプログラムになると、自分の思いもよらぬ箇所で、変数が変更されるなど不具合を起こしやすい。本当にグローバル変数が必要な場合は殆ど無い。）

変数のスコープの例

```
#include "stdafx.h"
int b = 1;  /* この変数 b はグローバル変数なので
            どこでも使える。*/

void f(int a){
    a = a+ 2; /* この変数 a はこの関数 f 内でのみ有効: mainのaとは無関係 */
     printf("in f() : a = %d¥n", a);
     printf("in f() : b = %d¥n", b);
 }
int _tmain(int argc, _TCHAR* argv[]){
      int a=1; /* この変数 a はこの関数 main 内でのみ有効: f のaとは無関係 */
      printf("in main() : a = %d¥n", a);
      f(a);
      printf("in main() : a = %d¥n", a);
      printf("in f() : b = %d¥n", b);
    getchar();
}
```

プログラムの実行結果

```
in main() : a = 1
in f() : a = 3
in f() : b = 1
in main() : a = 1
in f() : b = 1
```

課題

- 以下の関数を作って、計算させて下さい。

  配列の中で最大値を返す関数　int dataMax(int a[ ], int b) { }

  配列の中で最小値を返す関数　int dataMin(int a[ ], int b) { }

  配列の値の合計を返す関数　int dataSum (int a[ ], int b) { }

プログラム参考例（これを拡張して完成させて下さい）

```
#include "stdafx.h "
//ヒント:ここに関数を定義する
int _tmain(int argc, _TCHAR* argv[]){
    int i;   float  a[5];
    dataSet( a, 5);
    printf( "Max= %d ", dataMax( a , 5) );
    printf(" Min= %d ", dataMin( a , 5) );
   printf(" Sum= %d ", dataSum( a , 5) );
}
```

プロジェクト名：　useFunc

（このテキストの 関数に引数として配列を渡す例　を参考にして下さい）