

2021

計算機プログラミング演習

第14回 ユーザ定義関数と周辺

- 構造体 -

第14回の目標

- ユーザ定義関数のあれこれ
 - 戻り値(出力)が1つでない場合
 - 戻り値の型 (`char`, `int`, `double`, ...) の混在
 - 配列では対処できない
 - 引数の数が増大する
 - 単一の変数 (`char`, `int`, `double`...) ではなく変数のグループで取り扱いできないか？
- 構造体の使い方
 - 定義方法
 - 分割コンパイルとヘッダファイルによる定義の共通化

ユーザ定義関数(1)

- C言語の関数は数学関数と考え方が同じ
- 数学で使われる関数では

$$y = f(x)$$

f : 関数

x : 原因(入力)

y : 結果(出力)

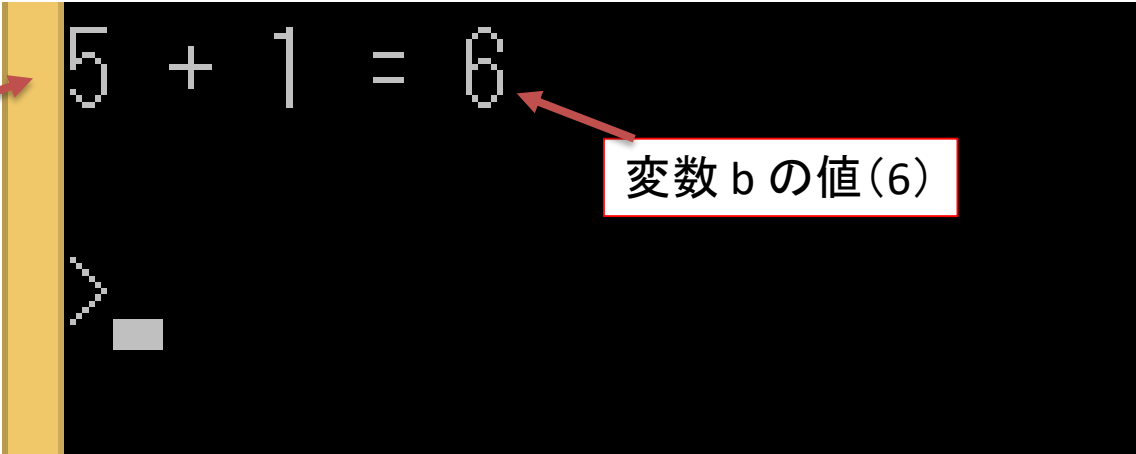
– 関数 $f(x)$ の演算の後に x が変化することはない

ユーザ定義関数(2)

```
// func_add.c :  
#include <stdio.h>  
  
int ftest(int);    /* プロトタイプ宣言 */  
  
int main( void )  
{  
    int b, a = 5;  
  
    b = ftest(a);  
    printf("%d + 1 = %d\n", a, b);  
    return 0;  
}  
  
int ftest(int a){  
    a = a + 1;  
    return a ;  
}
```

ユーザ定義関数(3)

- 実行結果



```
5 + 1 = 6
> █
```

変数 a の値 (5)

変数 b の値 (6)

引数と戻り値 (1)

- ユーザ定義関数の引数に使われる変数を関数内で変更しても呼び出し側には反映されない。
 - 関数内の変数(引数も含む)は呼び出し側の変数と独立している。
 - 引数は関数開始時に呼び出し側の引数の値を関数側の引数にコピーしている。
 - 関数終了時に関数側の引数は呼び出し側の引数には反映されない(コピーされない)。
 - 関数の出力は戻り値である。

引数と戻り値(2)

```
// func_add.c :  
#include <stdio.h>
```

```
int ftest(int); /* プロトタイプ宣言 */
```

```
int main( void )  
{
```

```
    int b, a = 5;
```

```
    b = ftest(a);  
    printf("%d + 1 = %d\n", a, b);  
    return 0;
```

```
}
```

```
int ftest(int a){  
    a = a + 1;  
    return a ;
```

```
}
```

変数 a の値(5)

呼び出し側とは独立した変数 a を作成し、値(5)を a にコピーする

変数 a の変更は引数に反映されない

return の後の変数 a の値は戻り値として呼び出し側に返される(コピー)

戻り値が1つでない場合を考える

- 関数の戻り値は一つと決められている
 - 関数定義部に関数の戻り値の型が定義されている(基本的に戻り値は一つ)
 - 戻り値を2つにしたい
 - 次のユーザ定義関数 `iaverage()` の戻り値を平均値と合計の2つにしたい
 - どうすれば良い？

ユーザー定義関数 iaverage()

```
#include <stdio.h>
```

```
int iaverage( int,  int [] ) ;
```

プロトタイプ宣言

```
int main( void )
```

```
{
```

```
    /* 2つの配列の平均を求める */
```

```
    :
```

```
    ave1 = iaverage( n, data1[] );
```

```
    :
```

```
    return 0;
```

```
}
```

戻り値はintと
なっている

```
int iaverage(int num, int data[]) {
```

```
    :
```

```
    :
```

```
    return ave;
```

```
}
```

関数本体

main() 部

```
int main( void )  
{
```

```
    /* 2つの配列の平均を求める */
```

呼び出し側と関数内部とで
変数名が異なっても構わない

```
    int data1[8] = { 45, 64, 58, 60, 51, 76, 45, 39 };  
    int data2[6] = { 162, 178, 172, 182, 171, 168 };  
    int ave1, ave2;
```

```
    ave1 = iaverage(8, data1);
```

```
    ave2 = iaverage(6, data2);
```

```
    printf("Average1=%d, Average2=%d\n", ave1, ave2);
```

```
    return 0;
```

```
}
```

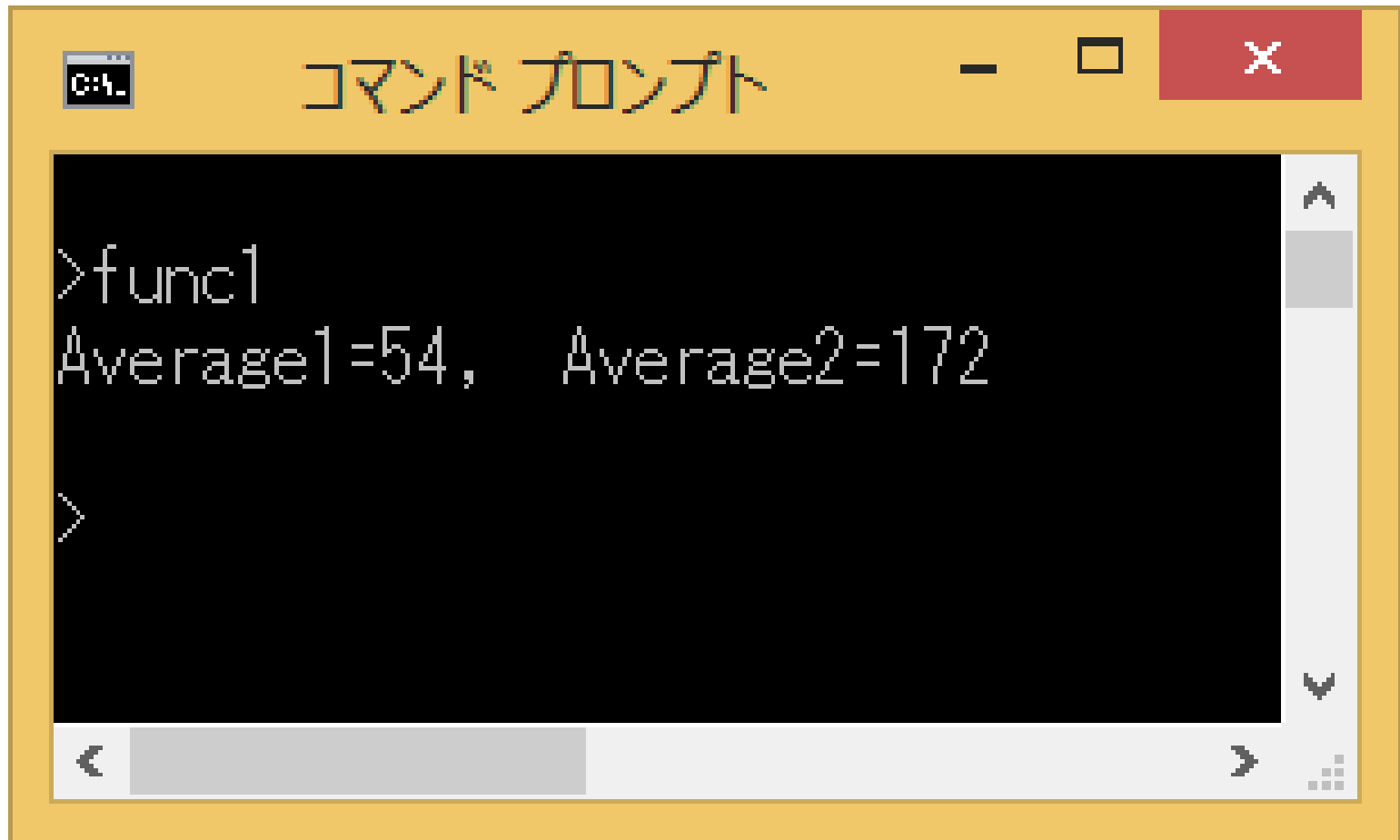
iaverage() 関数部

```
int iaverage(int num, int data[]) {  
    int sum, ave;  
    int i;  
  
    sum = 0;  
    for (i = 0; i < num ; i++){  
        sum = sum + data[i];  
    }  
  
    ave = sum / num ;  
  
    return ave;  
}
```

戻り値として平均値だけでなく合計も返したい

return は 関数を呼び出したプログラムに「戻る」

実行結果



A screenshot of a Windows Command Prompt window. The title bar is yellow and contains the text "コマンド プロンプト" (Command Prompt) in black. To the left of the title bar is a small icon of a command prompt window, and to the right are standard window control buttons (minimize, maximize, close). The main area of the window is black with white text. The text displayed is:
>func1
Average1=54, Average2=172
>
At the bottom of the window is a horizontal scrollbar with a grey track and a white slider.

```
>func1  
Average1=54, Average2=172  
>
```

関数の引数に構造体を使う

関数の引数の問題点

- プログラムの改良に従い、引数が増える
 - 関数の引数の数を増やした場合、以前（増やす前）のプログラムが動作しなくなる（後方互換性）
 - プログラムが複雑化すると、爆発的に引数の数が増加することがある
 - 変数の型や順番を整えなければならないので大変

構造体を使う

- 変数のグループを作る (構造体)
 - 複数の変数をいくつかまとめて、一つの変数のように扱う。
 - 配列と考えが似ているが、異なった型の変数が混在していても良い。
 - まとめる変数は配列であっても、構造体であっても構わない
 - ただし、構造体内部の**変数名** (メンバー変数) と **型**、**順番**などの定義は関数側も呼び出し側も共通でなければならない。
 - 呼び出し側と関数を別ファイルに分割している場合は重要

構造体の定義

```
struct 構造体型名 {  
    変数の型 変数名 ;  
    変数の型 変数名 ;  
    :  
};
```

構造体型 stoch の定義

```
struct stoch {  
    int sum ; /* 合計 */  
    int ave ; /* 平均 */  
};
```

```
int ave1, sum1 ;  
struct stoch st1 ;
```

stoch 型の構造体 st1 を作成

```
st1 = iaverage( n, data1[] ) ;  
ave1 = st1.ave ;
```

構造体 st1 のメンバー変数 ave
をアクセスする

構造体を使うプログラムに変更

```
// func1.c : 関数の使用法
#include <stdio.h>
struct stoch{
    int sum;
    int ave;
};

struct stoch iaverage(int, int[]); /* プロトタイプ宣言 */

int main( void )
{
    /* 2つの配列の平均を求める */
    int data1[8] = { 45, 64, 58, 60, 51, 76, 45, 39 };
    int data2[6] = { 162, 178, 172, 182, 171, 168 };
    struct stoch st1, st2;

    /* ユーザ定義関数 iaverage() を使う */
    st1 = iaverage(8, data1);
    st2 = iaverage(6, data2);

    printf("Sum1=%d, Average1=%d\n", st1.sum, st1.ave);
    printf("Sum2=%d, Average2=%d\n", st2.sum, st2.ave);
    return 0;
}
```


iaverage.c の変更

```
/* iaverage.c */

struct stoch{
    int sum;
    int ave;
};

struct stoch iaverage(int num, int data[]) {

    struct stoch st;
    int i ;

    st.sum = 0 ;
    for ( i = 0 ; i < num ; i++ ) {
        st.sum += data[ i ];
    }
    st.ave = st.sum / num;
    return st;
}
```

実行結果

 Command Prompt

```
>func1  
Sum1=438,   Average1=54  
Sum2=1033,  Average2=172  
  
>_
```

構造体定義を共通にする

- 構造体の定義を呼び出し側と関数側で共通にしなければならない
- 構造体の内容が複雑になったり、使用する範囲が大きくなったりすると、構造体の定義を共通に保つことが難しくなる
- 構造体定義の内容を追加、変更すると、使われているすべての関数の定義部も同様に変更しなければならない
 - バグのもと

ヘッダファイル (.h)を使って共有

- ヘッダファイルとは、複数の互にソースプログラムに共通部分を参照するためのファイル
- 共通部分を .h にまとめておき、必要なファイル(メインプログラム、関数ファイル)でインクルード(include)する
 - プログラム冒頭部の“ #include ” でファイルの中身をその場所に展開する
 - ユーザが作成したヘッダファイルはメインプログラムと同じディレクトリに置き、ヘッダファイル名を" " (ダブルクォート)で囲む

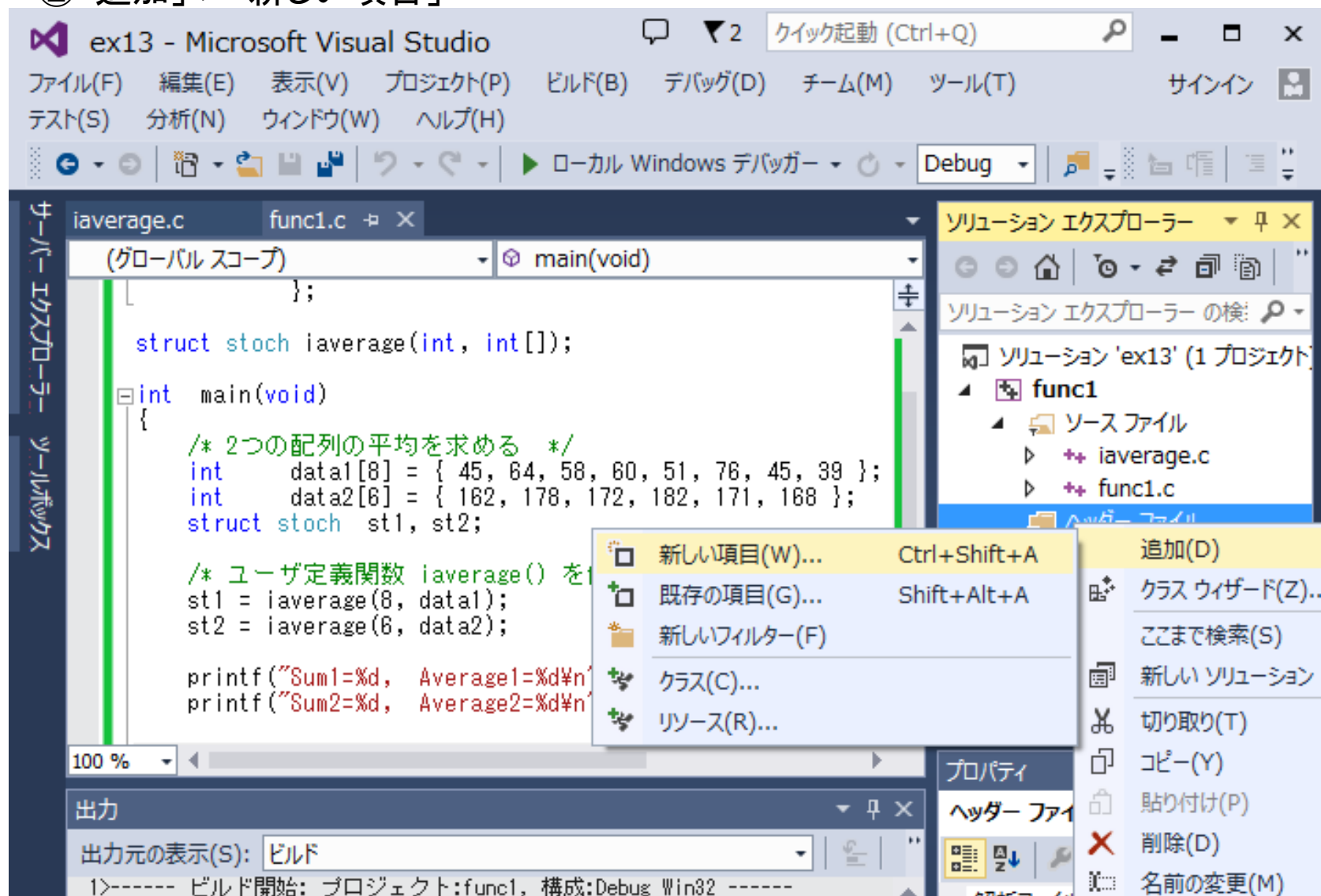
```
#include "stoch.h"
```
 - システムで用意されているヘッダファイルを使うときには <> で囲む

```
#include <stdio.h>
```
- 基本としてユーザが作成したヘッダファイル(*.h) はソースプログラムと同じディレクトリに置くこと

ヘッダファイルの作成(1)

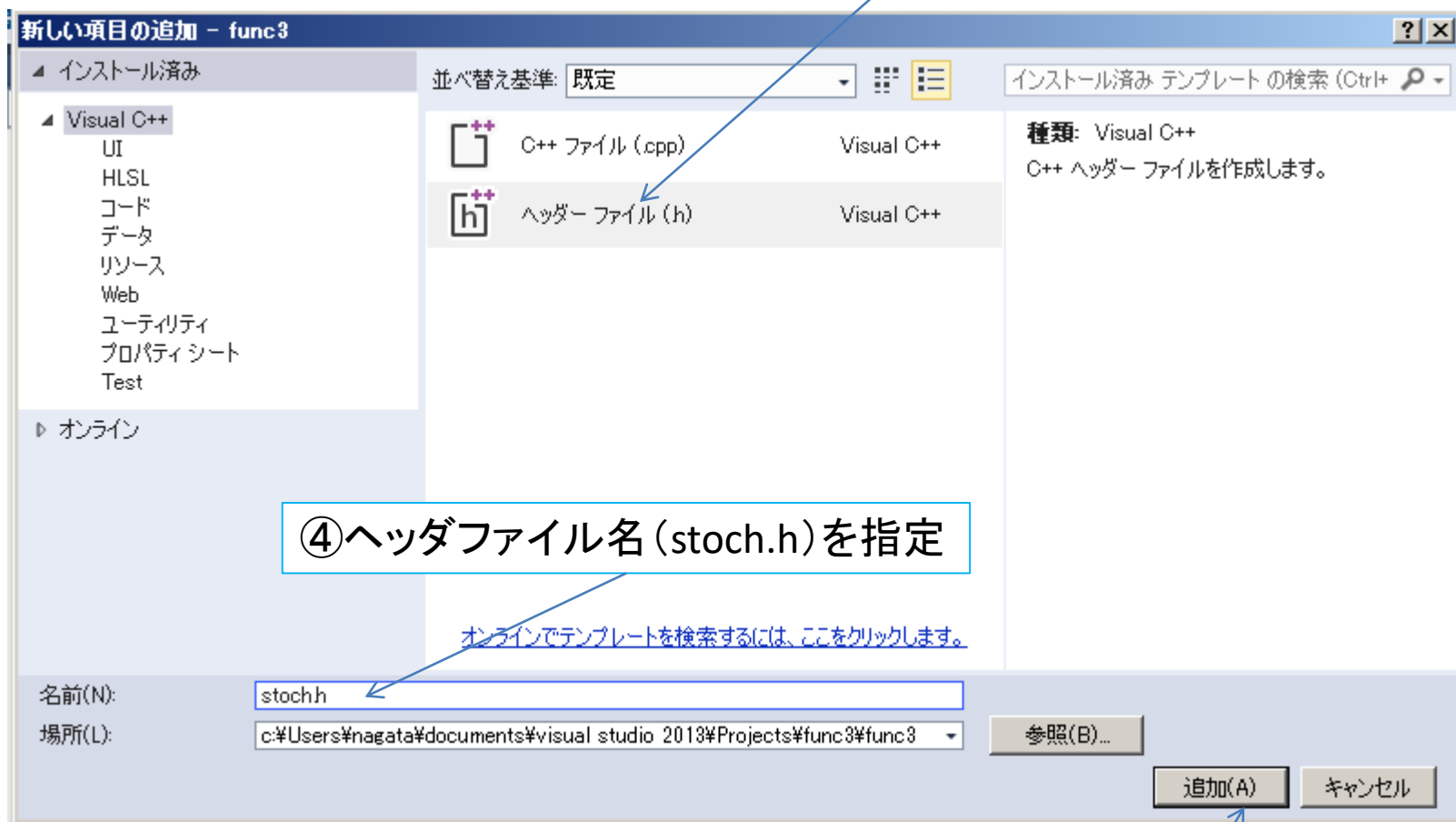
①画面右「ソリューション エクスプローラ」の「ヘッダーファイル」を右クリック

②「追加」->「新しい項目」



ヘッダファイルの作成(2)

③「ヘッダファイル (.h)」を選択



④ヘッダファイル名 (stoch.h)を指定

[オンラインでテンプレートを検索するには、ここをクリックします。](#)

⑤「追加」を押す

ヘッダファイルの作成(3)

stoch.h の内容 構造体型 stoch の定義のみ

```
/*stoch 構造体の定義 */  
  
struct stoch {  
  
    int    sum;           /* 合計 */  
    int    ave;          /* 平均 */  
  
};
```

func1.c , iaverage.c に共通する定義部を一つにまとめる。

前のプログラムのように、両方のプログラムに定義部がそれぞれあってもビルド、実行できるが、構造体内の変数などが変更になった場合に関連するソースプログラムをすべて書き換えなければならない。

そこで定義部を一つのファイルにして、各ソースプログラムから参照することにする。

func1.cの変更

```
// func1.c :
```

```
#include <stdio.h>
```

```
#include "stoch.h"
```

```
struct stoch iaverage(int, int[]);
```

```
int main( void )
```

```
{
```

```
    /* 2つの配列の平均を求める */
```

```
    int    data1[8] = { 45, 64, 58, 60, 51, 76, 45, 39 };
```

```
    int    data2[6] = { 162, 178, 172, 182, 171, 168 };
```

```
    struct stoch st1, st2;
```

```
        :  
        :  
        :
```

```
    return 0;
```

```
}
```


func1.c の変更

// func1.c : **BEFORE**

```
#include <stdio.h>
```

```
/* stoch 構造体の定義 */
```

```
struct stoch {
```

```
    int    sum;    /* 合計 */
```

```
    int    ave;    /* 平均 */
```

```
};
```

```
struct stoch iaverage(int, int[
```

```
int main( void )
```

```
{
```

```
    /* 2つの配列の平均を求める
```

```
    int    data1[8] = { 45
```

```
    int    data2[6] = { 162
```

```
    struct stoch st1, st2;
```

// func1.c : **AFTER**

```
#include <stdio.h>
```

```
#include "stoch.h"
```

```
struct stoch iaverage(int, in
```

```
int main( void )
```

```
{
```

```
    /* 2つの配列の平均を求める
```

```
    int    data1[8] = { 45
```

```
    int    data2[6] = { 162
```

```
    struct stoch st1, st2;
```

```
    :
```

```
    :
```

```
#include "ファイル名"
```

```
}
```

(ソースプログラムと同一のフォルダにあるヘッダファイルを指定する時)

iaverage.c の変更

/* iaverage.c **BEFORE** */

```
struct stoch{
    int sum;
    int ave;
};

struct stoch iaverage(int num, int
    int sum, ave, i;
    stoch st;

    sum = 0;
    for (i = 0; i < num; i++){
        sum = sum + data[i];
    }
    ave = sum / num;

    st.sum = sum;
    st.ave = ave;

    return st;
}
```

/* iaverage.c **AFTER** */

```
#include "stoch.h"
```

```
struct stoch iaverage(int
    int sum, ave, i;
    stoch st;

    sum = 0;
    for (i = 0; i < num;
        sum = sum +

    }
    ave = sum / num;

    st.sum = sum;
    st.ave = ave;

    return st;
}
```

よく使われる構造体の定義

```
typedef struct [構造体型名※] {  
    変数の型    変数名    ;  
    変数の型    変数名    ;  
    :  
} 型名    ;
```

※構造体型名は省略可

型 stc の定義

```
typedef struct    stoch {  
    int    sum    ; /* 合計 */  
    int    ave    ; /* 平均 */  
} stc;
```

```
int    ave1, sum1 ;  
stc    st1    ;
```

stc 型の構造体 st1 を作成

```
st1 = iaverage( n, data1[] ) ;  
ave1 = st1.ave    ;
```

構造体 st1 のメンバー変数 ave
をアクセスする

ヘッダファイル stoch.h の変更

stoch.h の内容

構造体型 stc の定義

```
/*stoch 構造体の定義 */  
  
typedef struct stoch {  
  
    int    sum;           /* 合計 */  
    int    ave;          /* 平均 */  
  
} stc ;
```

func1.c , iaverage.c に共通する定義部を一つにまとめる。

前のプログラムのように、両方のプログラムに定義部がそれぞれあってもビルド、実行できるが、構造体内の変数などが変更になった場合に関連するソースプログラムをすべて書き換えなければならない。

そこで定義部を一つのファイルにして、各ソースプログラムから参照することにする。

func1.c

```
// func1.c : 関数の使用法
#include <stdio.h>

#include "stoch.h"

stc iaverage(int, int[]); /* プロトタイプ宣言 */

int main( void )
{
    /* 2つの配列の平均を求める */
    int data1[8] = { 45, 64, 58, 60, 51, 76, 45, 39 };
    int data2[6] = { 162, 178, 172, 182, 171, 168 };
    stc st1, st2;

    /* ユーザ定義関数 iaverage() を使う */
    st1 = iaverage(8, data1);
    st2 = iaverage(6, data2);

    printf("Sum1=%d, Average1=%d¥n", st1.sum, st1.ave);
    printf("Sum2=%d, Average2=%d¥n", st2.sum, st2.ave);

    return 0;
}
```

次スライドにつづく

つづき

iaverage.c (isum()) 関数)

```
/* iaverage.c */
#include "stoch.h"

int isum(int num, int data[]) {
    int i, sum = 0 ;
    for ( i = 0 ; i < num ; i ++ ) {
        sum += data[ i ] ;
    }
    return sum ;
}

stc iaverage(int num, int data[]) {

    stc st;

    st.sum = isum( num, data ) ;
    st.ave = st.sum / num;

    return st;
}
```