

PROVIT: 初学者向けC言語 オンライン教育支援ツール

公立法人 会津大学

原 昂平

PROVITとは

- ◆ **PRO**gramming **V**isualization **T**oolの略
- ◆ 任意のCソースコードを**逐次実行**できる
- ◆ 実行過程を**可視化**して利用者に見せる
- ◆ **オンライン**で使用できる
 - ◆ <http://cleast.u-aizu.ac.jp/>

従来の授業

- ◆ 板書やスライドを使った講義形式
 - ◆ プログラムの流れを説明しにくい
 - ◆ 変数の値の移り変わりを表現しにくい
 - ◆ 繰り返し説明するのに向いていない



PROVITでの授業

- ◆ キーボード操作でプログラムの動きを見せることが可能
- ◆ スライドから起動することが可能
- ◆ 何度でも同じプログラムを動かすことができる



PROVIT利用環境

- ◆ Java最新版
- ◆ インターネット環境
- ◆ ブラウザ
 - ◆ Windows, Mac, Linuxそれぞれの推奨ブラウザ

PROVITの起動

- ◆ <http://cleast.u-aizu.ac.jp/>にアクセスします

The screenshot shows the PROVIT website interface. At the top left, there is a code editor window displaying C code:

```
return 0;
}
```

 Below it, a terminal window shows the output:

```
main()
11
```

 To the right of the code editor, there is a text box:

マウスホイールで拡大・縮小
マウスドラッグで移動

 On the right side of the page, there are several navigation links:

[ご利用の前に](#)
[左の例を動かす](#)
[自由利用](#)
[PowerPointでの利用例](#)

 Below these links, there is a section titled **PROVITを利用した講義風景** with a small image of a computer monitor displaying `main()`. At the bottom, there is a list of lecture examples under the heading **PROVITで学習する例題集**:

- [Cプログラミング超入門 パソコン甲子園で点を取ろう\(工事中\)](#)
- [会津大学1年前期授業 プログラミング入門](#)
- [会津大学1年後期授業 プログラミングC\(工事中\)](#)

自分で書いたプログラムを試す時はこちらをクリック

[自由利用](#)

授業のサンプルを動かす時はこちらをクリック

[会津大学1年前期授業 プログラミング入門](#)

オンラインサンプルの起動

PROVITでCプログラミング 例題集
会津大学 プログラミング入門

会津大学1年前期授業「プログラミング入門」で使用されるCプログラムです。対応するハンドアウトは会津大学売店で購入できます。

回	授業内容	sample program	スライド番号	ハンドアウトのページ番号
1	コンピュータとは～Helloの表示	Helloと表示するプログラム(lec01-1.c)	22	6
		改行表示と出力の仕方の違い(lec01-2.c)	23	6
		1+2を計算するプログラム(lec01-3.c)	24	6
2	変数・入出力・代入・演算a	1+2を計算するプログラム(復習)(lec01-3.c)	8	8
		1+2=3を出力するプログラム(lec02-1.c)	13	10
		scanfの使用例(lec02-2.c)	23	12
		プログラム例 (平均を求める) (lec02-3.c)		12
3	変数・入出力・代入・演算b	混合演算による型変換時の注意：割り算(lec03-1a.c)	10	
		キャスト - 明示的な型変換(lec03-1b.c)	11	15
		printfの書式(lec03-2.c)	13	16
		printfの高度な書式(整数)(lec03-3a.c)	14	16
		printfの高度な書式(浮動小数点)(lec03-3b.c)	15	16
		プログラム例 scanf/printfの書式(lec03-4.c)	18	17
		プログラム例 平均を求める(lec03-5.c)	19	17
	選択(if文)(lec04-1.c)	6	19	

クリック

プログラム作成・変更

プログラムチェック (コンパイル・実行・解説)

PROVITを評価

情報

プログラムを書いたら
このタブで実行できます

このタブでプログラムを
書いたり直したりできます

保存してあるプログラムを
開くことができます

ここで書いたプログラムを
保存できます

読込
保存



プログラム作成・変更 プログラムチェック (コンパイル・実行・解説) PROVITを評価 情報

+(Q) -(A)

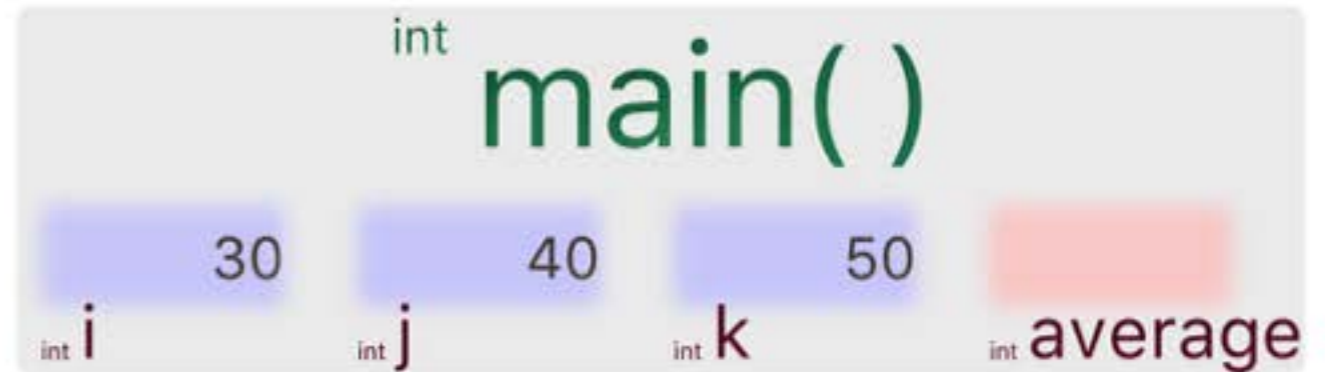
```

2  * lec02-3.c
3  * 3人分の体重を入力し、平均を求める
4  *
5  */
6
7  #include <stdio.h>
8  int main()
9  {
10 int i,j,k,average;
11
12 printf("3人分の体重(整数kg)を空白で分けて入力してください：")
13 scanf("%d%d%d",&i,&j,&k);
14
15 average = (i + j + k)/3;
16
17 printf("3人の体重はそれぞれ %dkg %dkg %dkgで、\n")
18 printf("その平均は %dkg となりました\n",average);
19 return 0;
20 }

```

書いたプログラムが
表示されます

+(P) -(L)



変数の値の移り変わりを
ここで確認できます

Console scanf()等の、標準入力関数の入力先設定

3人分の体重(整数kg)を空白で分けて入力してください： 30 40 50

scanf()とprintf()は
ここで入力(出力)します

戻り(B)

ステップ進み(N)

連続進み(G)

最初に戻る(R)

プログラム作成・変更 プログラムチェック (コンパイル・実行・解説) PROVITを評価 情報

+(Q) -(A)

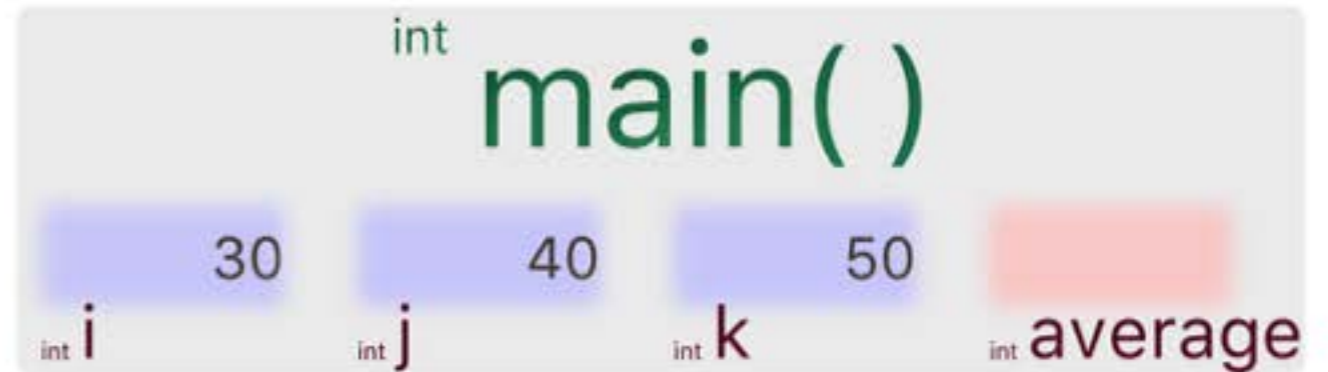
```

2  * lec02-3.c
3  * 3人分の体重を入力し、平均を求める
4  *
5  */
6
7  #include <stdio.h>
8  int main()
9  {
10 int i,j,k,average;
11
12 printf("3人分の体重(整数kg)を空白で分けて入力してください：")
13 scanf("%d%d%d",&i,&j,&k);
14
15 average = (i + j + k)/3;
16
17 printf("3人の体重はそれぞれ %dkg %dkg %dkgで、\n",i,j,k);
18 printf("その平均は %dkg となりました\n",average);
19 return 0;
20 }

```

ドラッグでスクロール
できます

+(P) -(L)



ドラッグで動かせます

Console scanf()等の、標準入力関数の入力先設定

3人分の体重(整数kg)を空白で分けて入力してください： 30 40 50

戻り(B)

ステップ進み(N)

連続進み(G)

最初に戻る(R)

プログラム作成・変更

プログラムチェック (コンパイル・実行・解説)

PROVITを評価

情報

+(Q) · -(A)

+(P) · -(L)

2 lec02-3.c

3 * 3人分の体重を入力し、平均を求める

4 *

5 *

6 *

7 #include <stdio.h>

8 int main()

9 {

プログラムの拡大/縮小を
することができます

14

15 average = (i + j + k)/3;

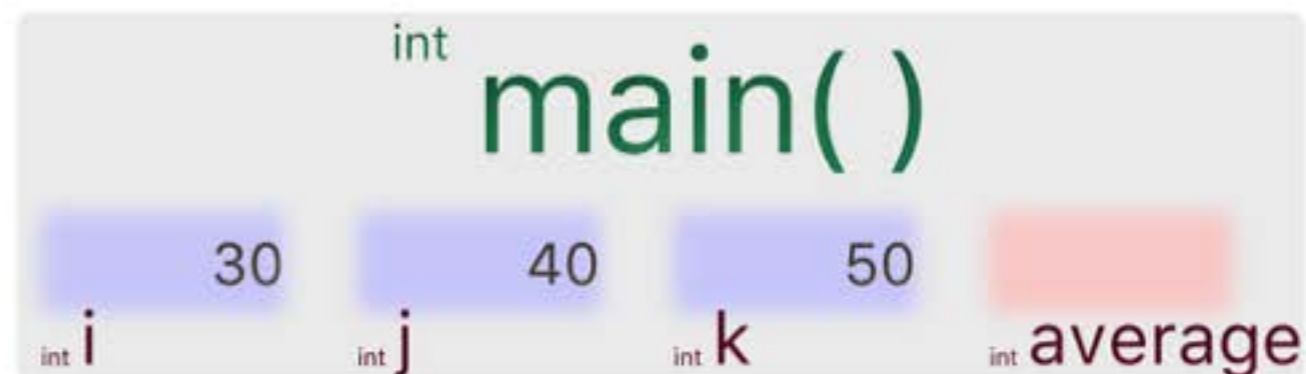
16

17 printf("3人の体重はそれぞれ %dkg %dkg %dkgで、\n",i,j,k);

18 printf("その平均は %dkg となりました\n",average);

19 return 0;

20 }



入力してください:'

箱の拡大/縮小を
することができます

Console scanf()等の、標準入力関数の入力先設定

3人分の体重(整数kg)を空白で分けて入力してください: 30 4

戻り(B)

ステップ進み(N)

連続進み(G)

最初に戻る(R)

プログラム作成・変更

プログラムチェック (コンパイル・実行・解説)

PROVITを評価

情報

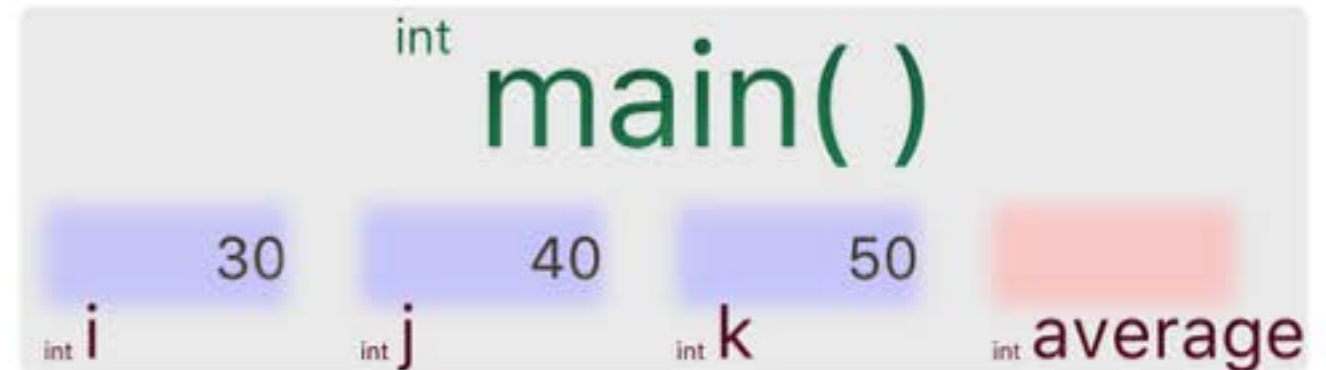
+(Q) - (A)

+(P) -(L)

```

2 * lec02-3.c
3 * 3人分の体重を入力し、平均を求める
4 *
5 */
6
7 #include <stdio.h>

```



動作を1つ戻します

動作を1つ進めます

最後まで動作を進めます

動作を最初に戻します

```

10 int i, j, k, average;
11
12 printf("3人分の体重(整数kg)を空白で分けて入力してください:");
13 scanf("%d%d%d",&i,&j,&k);
14
15 average = (i+j+k)/3;
16
17 printf("3人の体重はそれぞれ %dkg %dkg %dkgで、\n",i,j,k);
18 printf("その平均は %dkg となりました\n",average);
19 return 0;
20 }

```

の、標準入力関数の入力先設定

3人分の体重(整数kg)を空白で分けて入力してください: 30 40 50

戻り(B)

ステップ進み(N)

連続進み(G)

最初に戻る(R)

変数の表示

```

2 * lec03-1b.c
3 * calculation with cast
4 * (correct version)
5 */
6 #include <stdio.h>
7
8 int main()
9 {
10 int i = 10;
11 int j = 4;
12 double a;
13
14 a = (double)i / (double)j; /* キャストは分子/分母片方だけでも良い */
15 printf("result: %f\n",a);
16 return 0;
17 }

```

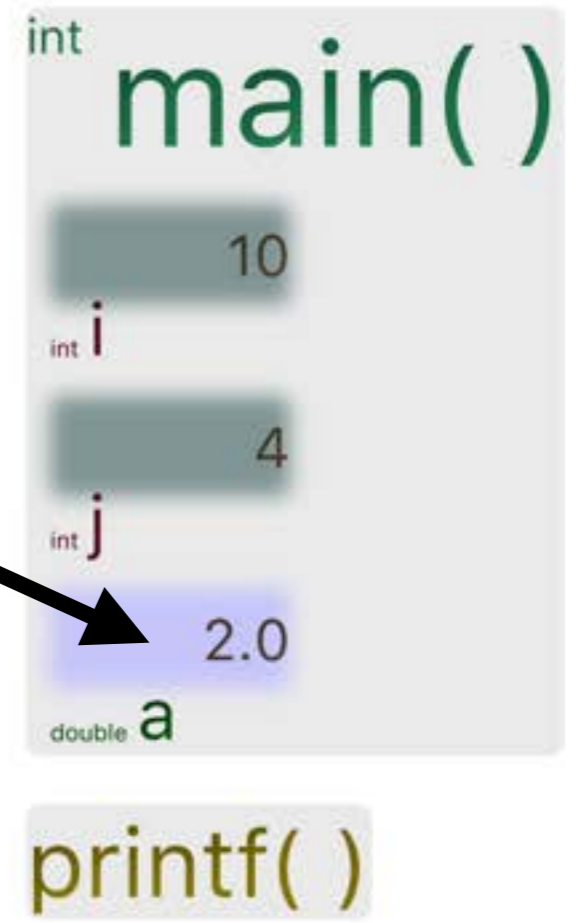
変数の型と名前と値を
一緒に確認できます



+(Q) : -(A)

```
2 * lec03-1b.c
3 * calculation with cast
4 * (correct version)
5 */
6 #include <stdio.h>
7
8 int main()
9 {
10 int i = 10;
11 int j = 4;
12 double a;
13
14 a = i / j; /* キャストは分子/分母片方だけでも良い */
15 printf("result : %f\n",a);
16 return 0;
17 }
```

計算やキャストが間違ったときにすぐに確認できます



Console scanf()等の、標準入力関数の入力先設定

```
6 #include <stdio.h>
7
8 int main()
9 {
10 int i = 10;
11 int j = 4;
12 double a;
13
14 a = (double)1 / (double)4;
15 printf("result : %f\n",a);
16 return 0;
```

変数のデータ型 (8バイト実数型)

右クリックでプログラムの説明が表示されます

分岐

if文

```
7 #include <stdio.h>
8
9 int main()
10 {
11     int signal;
12
13     printf("0:red, 1:green, 2:yellow : ");
14     scanf("%d",&signal);
15
16     if(signal == 0) printf("Stop\n");
17     else if(signal == 1) printf("Go\n");
18     else if(signal == 2) printf("Be careful\n");
19     else printf("Look at the traffic signal\n"); /* あり得ない! */
20
21     return 0;
22 }
```

int
main()
1
int signal

青い下線でどこを通ったか
確認することができます

Console scanf()等の、標準入力関数の入力先設定
0:red, 1:green, 2:yellow : 1
Go

戻り(B) ステップ進み(N) 連続進み(G) 最初に戻る(R)

if文

```
13 printf("0:red, 1:green, 2:yellow : ");
14 scanf("%d",&signal);
15
16 if(signal == 0) printf("Stop\n");
17 else if(signal == 1) printf("Go\n");
18 signalが0なら青色部分の処理を実行 careful\n");
19 else printf("Look at the traffic signal\n");
20
21 return 0;
22 }
```

右クリックでプログラムの説明が表示されます

if文

```
if(a==0 && (b==1 || c==3) ) {
```

```
    printf("1");
```

以下の全ての条件が真なら青色部分の処理を実行

- aが0
- bが1またはcが3

```
if(a==0 && b==1 || c==3) {
```

```
if(a==0 && b==1 || c==3) {
```

```
    printf("2");
```

以下のいずれかの条件が真なら青色部分の処理を実行

- aが0かつbが1
- cが3

AND, ORの優先順位の
違いを確認できます

switch-case文

```
+ (Q) r - (A) main()
10 {
11 int signal;
12
13 printf("0:red, 1:green, 2:yellow : ");
14 scanf("%d",&signal);
15
16 switch(signal){
17 case 0:
18 printf("Stop\n");
19 break;
20 case 1:
21 printf("Go\n");
22 break;
23 case 2:
24 printf("Be careful\n");
25 break;
26 default:
27 printf("Look at the traffic signal\n"); /* あり得ない！ */
28 }
29
30 return 0;
31 }
```

```
int
main()
1
int signal
```

青い下線でどこを通ったか
確認することができます

```
Console scanf()等の、標準入力関数の入力先設定
0:red, 1:green, 2:yellow : 1一ボードより入力する
Go
█
```

switch-case文

```
16 switch(signal){
17 case 0:
18     printf("Go\n");
19     break;
20 case 1:
21     printf("Go\n");
22     break;
23 case 2:
24     printf("Traffic signal\n");
25     break;
26 case 3:
27     printf("Traffic signal\n");
28 }
29
```

signalの値に従い処理を振り分ける

右クリックでプログラムの説明が表示されます

switch-case文

break文の有無の違いが
はっきりと出ます

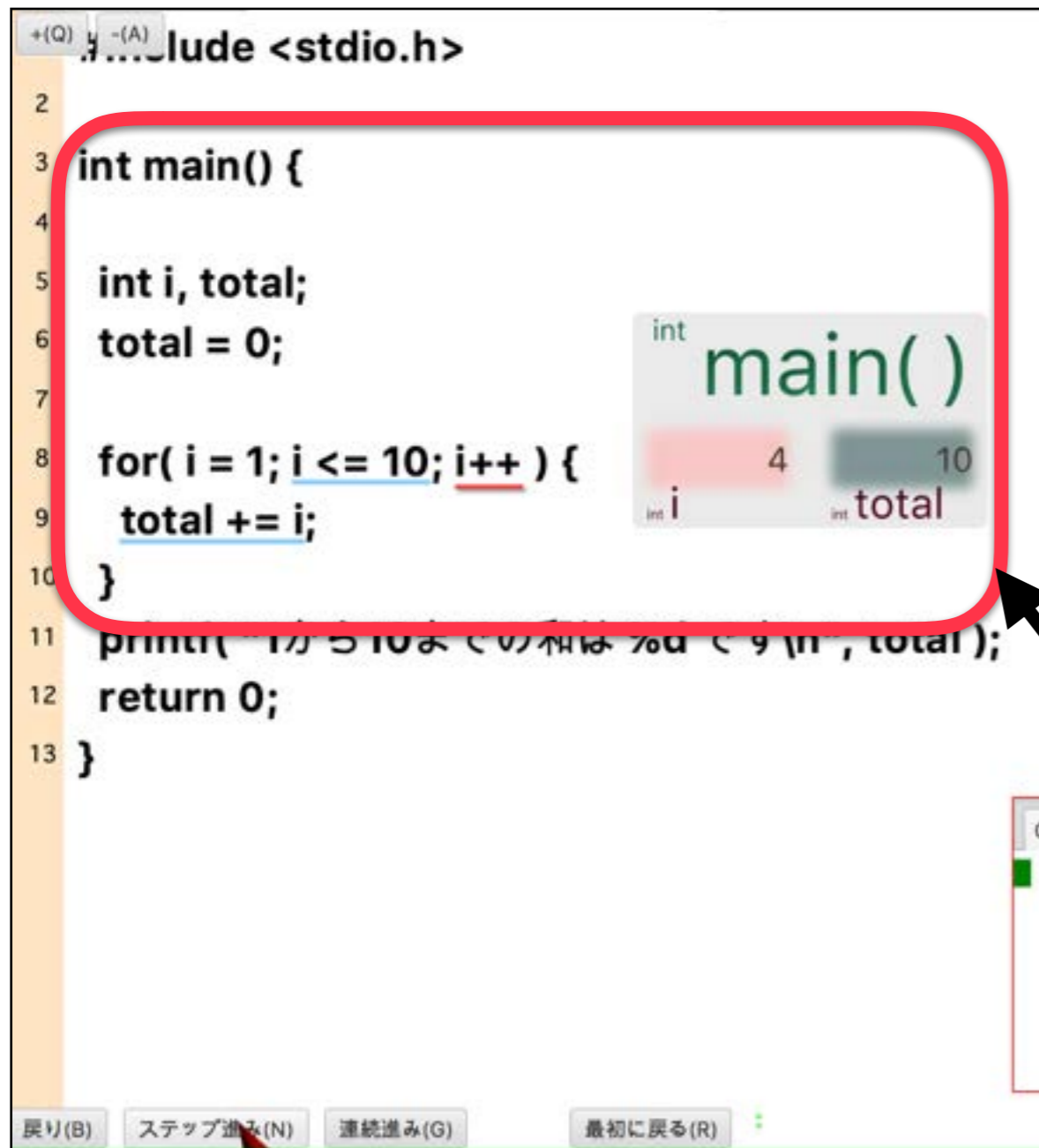
```
16 swit signalの値が0の場合青色部分の処理を実行
17 case 0:
18     printf("Stop\n");
19     break;
20 case 1:
21     printf("Go\n");
22     break;
23 case 2:
24     printf("Be careful\n");
25     break;
26 default:
27     printf("Look at the traffic signal\n");
28 }
```

```
16 sv signalの値が0の場合青色部分の処理を実行
17 case 0:
18     printf("Stop\n");
19 case 1:
20     printf("Go\n");
21 case 2:
22     printf("Be careful\n");
23 default:
24     printf("Look at the traffic signal\n"); /* あ
25 }
```

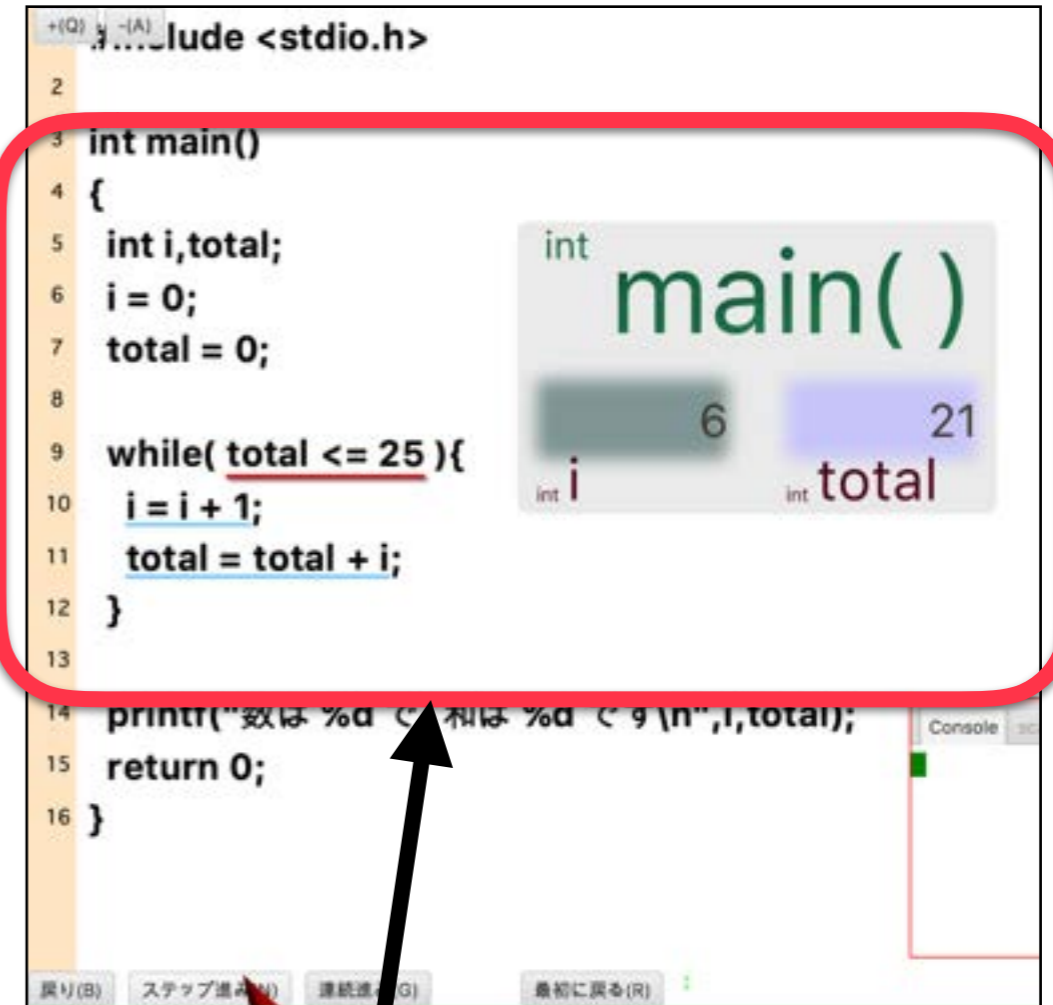
ループ

while文, for文

```
+ (Q) - (A) ... lude <stdio.h>
2
3 int main() {
4
5 int i, total;
6 total = 0;
7
8 for( i = 1; i <= 10; i++ ) {
9     total += i;
10 }
11 printf( "1から10までの和は %d です\n", total );
12 return 0;
13 }
```



```
+ (Q) - (A) ... lude <stdio.h>
2
3 int main()
4 {
5 int i, total;
6 i = 0;
7 total = 0;
8
9 while( total <= 25 ){
10     i = i + 1;
11     total = total + i;
12 }
13
14 printf( "数は %d の和は %d です\n", i, total );
15 return 0;
16 }
```



マウスやキーボード操作で
繰り返し中の変数の動きを
見ることができます

while文, for文

```
7 見張り方式ループ：  
8  totalが25以下の間に青色部分の処理を繰り返し実行  
9  while( total <= 25 ){  
10     i = i + 1;  
11     total = total + i;  
12 }
```

右クリックで
プログラムの説明が
表示されます

```
3  int main() {  
4  
5  int i, total;  
6  total = 0;  
7  
8  for( i = 1; i <= 10; i++ ) {  
9      total += i;  
10 }  
11 printf( "1から10までの和は %d です\n", total );  
12 return 0;
```

カウンタ方式ループ：
(1) [カウンタ初期化] iに1を代入
(2) [継続判断] 「iが10以下」が偽ならループ終了
(3) [青色部分の処理を一度実行]
(4) [カウンタ更新] iの値を1増やす
(5) (2)へ

前置と後置の違いも
説明を表示すると見やすい

```
8 j = ++i;  
9  
10 j = ++i;  
11  
12 j = i++;
```

iの値を1増やしてから
jにiを代入

```
10 j = ++i;  
11  
12 j = i++;  
13  
14 j = ++i;
```

jにiの値を代入
そのあとiの値を1増やす

1次元配列

1次元配列

The screenshot shows a development environment window titled "PROVIT-FX 2015. 会津大学 黒川研究室". The code editor contains the following C code:

```
1 #include <stdio.h>
2
3 main()
4 {
5     int i, man[10], sum = 0 ;
6
7     for(i = 0 ; i < 10 ; i++){
8         scanf("%d",&man[i]);
9         sum = sum + man[i];
10    }
11    printf("Average = %f\n", (double)sum/24.0);
12 }
```

Overlaid on the code is a diagram of a variable declaration for `main()`. It shows:

- `int i` with a value of 5.
- `int man[10]` with a value of 150. This declaration is highlighted with a red box, and a red mouse cursor points to it.
- `int sum` with a value of 150.

A black arrow points from a text box at the bottom to the `man[10]` declaration in the diagram.

配列は最初このような
状態で表示される

1次元配列

The image shows a screenshot of a C program editor. The code on the left is as follows:

```
1 #include <stdio.h>
2
3 main()
4 {
5     int i, man[10], sum = 0 ;
6
7     for(i = 0 ; i < 10 ; i++){
8         scanf("%d",&man[i]);
9         sum = sum + man[i];
10    }
11    printf("Average = %f\n", (double)sum/24.0);
12 }
```

Overlaid on the code is a variable viewer window titled "int man[10]: 配列". The window displays a table of values for the array:

10	20	30	40	50	0
(0)	(1)	(2)	(3)	(4)	(5)

Below the table, the variable viewer shows the state of variables in the current scope:

- int i: 5
- man[10]: (array)
- int sum: 150


An arrow points from the `scanf()` function call in the code to the variable viewer window.

クリックすると別ウィンドウで表示される

2次元配列

2次元配列

```
PROVIT-FX 2015. 会津大学 黒川研究室
プログラム作成・変更 プログラムチェック (コンパイル・実行・解説) PROVITを評価 情報
+(Q) : -(A) ude <stdio.h> +(P) -(L)
2
3 int main(){
4   int data[3][5], i, j;
5
6   for(i = 0; i < 3; i++){
7     for(j = 0; j < 5; j++){
8       scanf("%d", &data[i][j]);
9     }
10  }
11  return 0;
12 }
```



1次元配列同様、最初はこの状態で表示される

2次元配列

The screenshot shows a C program in a development environment. The code defines a 2D array `data` of type `int` with dimensions `3x5`. The program uses nested `for` loops to read values from the array. A separate window displays the array's state as a table, with the value `8` at index `(1,2)` highlighted in green. Below the table, a variable declaration for `main()` shows `data[3][5]` and indices `i` and `j`.

```
#include <stdio.h>

int main(){
    int data[3][5], i, j;

    for(i = 0; i < 3; i++){
        for(j = 0; j < 5; j++){
            scanf("%d", &data[i][j]);
        }
    }
    return 0;
}
```

1	2	3	4	5
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
6	7	8	0	0
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
0	0	0	0	0
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)

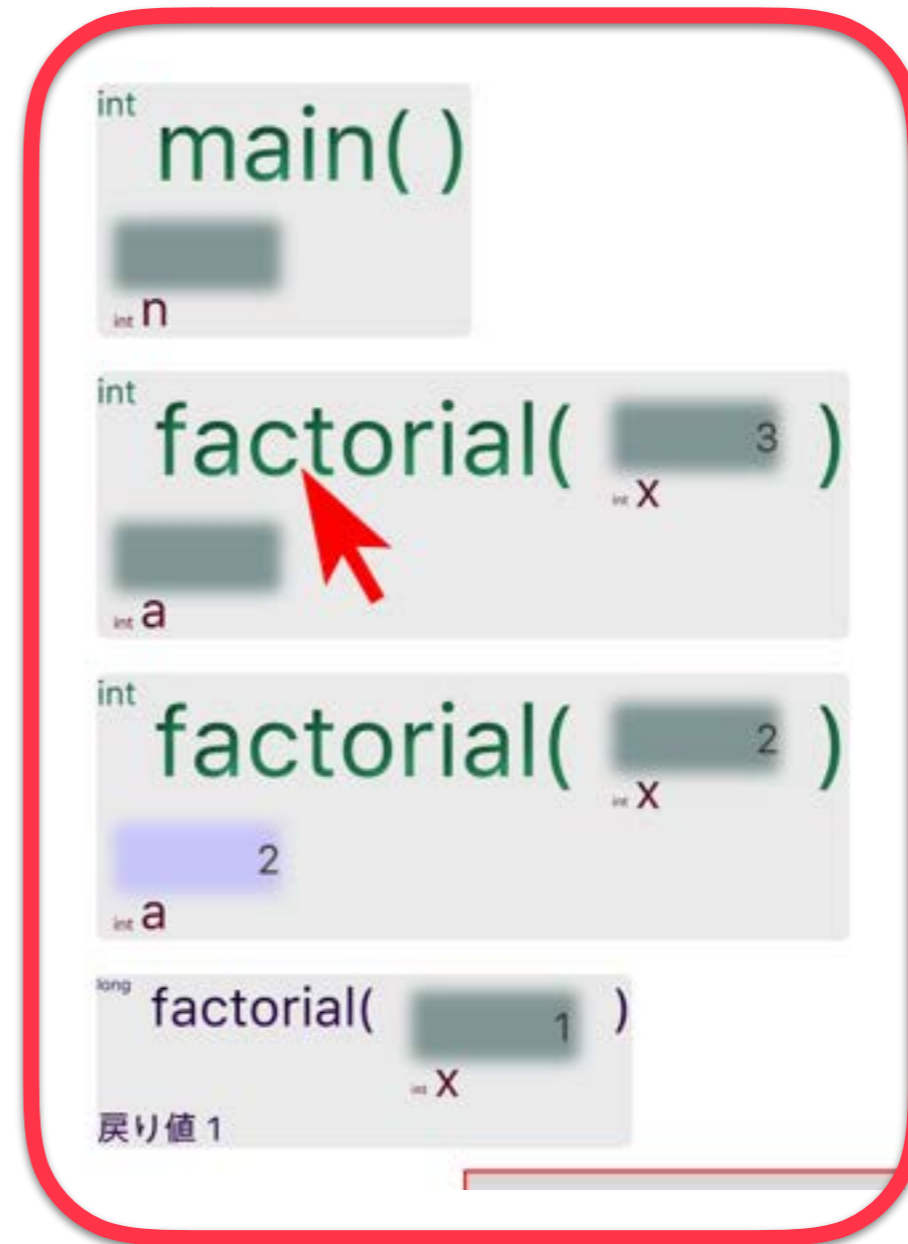
int main()
data[3][5] i j

1次元配列同様、別ウィンドウで表形式で表示される

関数

関数

```
5 int main(void){
6   int n;
7   n = factorial(3);
8   printf("n=%d\n",n);
9 }
10
11 int factorial(int x){
12   int a;
13   if(x==1) return 1;
14   a = x * factorial(x-1);
15   return a;
16 }
```



main()とは別の箱が表示される。
関数の処理が終わると戻り値が表示される

PROVITの将来性

- ◆ 会津大学の「プログラミング入門」の授業で実際に使用されている
- ◆ 現状では関連ファイルを手作業で用意する必要がある
- ◆ 学習者に合わせた表現に修正していくことでよりよい学習環境を学習者へ提供できる

ご意見お待ちしております
<http://cleast.u-aizu.ac.jp>