9. 配列

● 配列とは

プログラムの中で、複数のデータを扱う場合が数多くある。こうした場合には、数学の数列$(x_0, x_1, x_2, \cdots)$で用いた添え字付きの変数を使用する。

```
#include <stdio.h>
int main(void){
        int x[3];
        int i;
        x[0]=0;
        x[1]=1;
        x[2]=2;
        for(i=0;i<3;i++){
                printf("x[%d]=%d¥n");
        {
}
```

配列の宣言
配列に代入する値
添え字

図 1　配列サンプル 1

● 配列の役割

図 2 に示す「平均値を求めるプログラム」では、複数個の変数を用意し、この変数に値を入力する構文も、複数個用意する必要がある。

```
#include <stdio.h>
int main(void){
        int a,b,c,d,e;
        float heikin;

        printf("a=");
        scanf("%d",&a);
        printf("b=");
        scanf("%d",&b);
        printf("c=");
        scanf("%d",&c);
        printf("d=");
        scanf("%d",&d);
        printf("e=");
        scanf("%d",&e);

        heikin=(a+b+c+d+e)/5.0;

        printf("平均値=%lf¥n",heikin);
}
```

5 個の変数宣言
5 個の値を入力する構文

図 2　平均値を求めるプログラム

配列の添え字付きの変数を用いることによって、次ページの図 3 に示すように、複数の値を入力する(同一)構文を for 文で作成することができ、数行にまとめることができる。

```
#include <stdio.h>
int main(void){
        int data[5];        //平均を求める各値(5 個)
        int i;              //ループカウンタ
        int sum=0;          //加算値
        float heikin=0.0;   //平均値

        //各値の入力
        for( i=0; i<5; i++ ){
                printf( “data[%d]=”,i );
                scanf( “%d”,&data[i] );
        }

        //加算
        for( i=0; i<5; i++ )
                sum=sum+data[i];
        //平均値の計算
        printf( “平均値=%lf¥n”,heikin=sum/5.0 );
}
```

5 個の値を入力する構文(for 文使用)

図 3　平均値を求めるプログラム(配列使用)

● 配列の宣言と初期値の与え方

① 初期値を与えず、要素数(添え字数)を指定する

型名　配列名[要素数];

例：　int data[3];

② 初期値を与えて、要素数(添え字数)を指定する

型名　配列名[要素数]={初期値 0,初期値 1,初期値 2,…};

例：　int data[3]={1,2,3};

③ 初期値を与えて、要素数(添え字数)を指定しない

型名　配列名[ ]={初期値 0,初期値 1,初期値 2,…};

例：　int data[ ]={1,2,3};

※初期値の数だけ、要素数(添え字数)が用意される

● 配列への値の代入

① 添え字が数字の場合

data[1]=2;　　※1 番目の配列変数 data に値 2 を代入する

② 添え字が変数の場合

data[i]=2;　　※i 番目の配列変数 data に値 2 を代入する

③ 添え字が式の場合

data[i+1]=2;　　※i+1 番目の配列変数 data に値 2 を代入する

● 配列の式での利用

① a=data[1]+data[2];　　※1 番目と 2 番目の配列変数 data の値を加算して、変数 a に代入する

② b=data[i]+data[i+1];　　※i 番目と i+1 番目の配列変数 data の値を加算して、変数 b に代入する

● 配列の仕組み

配列を宣言すると、メモリに配列変数の型に応じた大きさの領域が確保される。int（整数）型では、1つの配列変数に対して 4byte(ﾊﾞｲﾄ)の領域ができる。

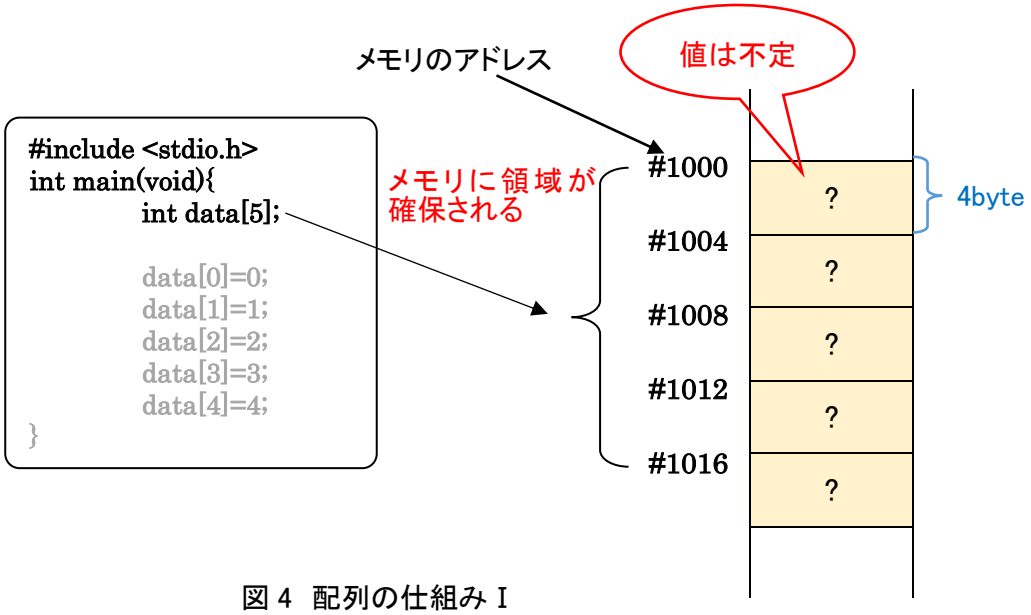

図 4　配列の仕組み Ⅰ

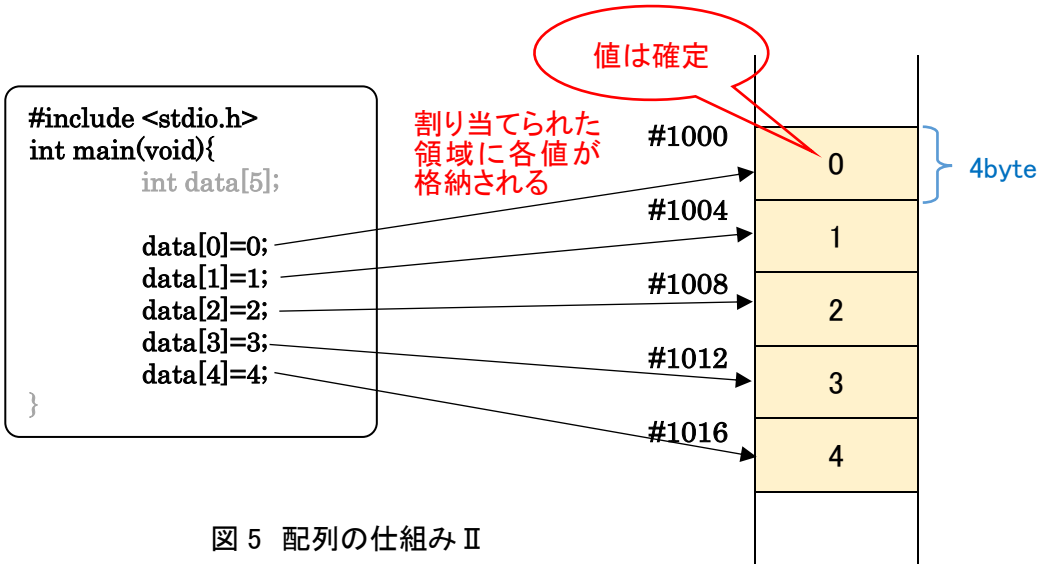

図 5　配列の仕組み Ⅱ

● 2 次元配列

配列では、数学の行列のように 2 次元で変数を扱うことができる

① 2 次元配列の宣言と初期値の与え方

- 型名 配列名[要素数][要素数];

例： int data[2][2];

- 型名 配列名[要素数][要素数] = { {初期値 00,初期値 01,… } , {初期値 10,初期値 11,… } , … };

例： int data[2][2]={ { 0,1 } , { 2,3 } }; ※data[0][0]=0 data[0][1]=1 data[1][0]=2 data[1][1]=3 が格納される

- 型名 配列名[ ][要素数] = { {初期値 00,初期値 01,… } , {初期値 10,初期値 11,… } , … };

例： int data[ ][2]={ { 0,1 } , { 2,3 } }; ※初期値の数だけ、要素数が用意される

- 型名 配列名[ ][ ] = { {初期値 00,初期値 01,… } , {初期値 10,初期値 11,… } , … };

例： int data[ ][ ]={ { 0,1 } , { 2,3 } }; ※初期値の数だけ、要素数が用意される

```
#include<stdio.h>
nt main(void){
        int data[][2]={{0,1},{2,3}};            //配列宣言と初期化
        int i,j;                                //添え字カウンタ

        printf( “2 次元配列 data¥n” );
        for( i=0;i<2;i++ ){                     //添え字 i のループ
                for( j=0;j<2;j++ ){             //添え字 j のループ
                        printf( “%d “,data[i][j] );     //配列要素の表示
                }
        printf( “¥n” );
        }
}
```

図 6 2 次元配列要素の表示プログラム Ⅰ

② 2 次元配列の要素と添え字の関係

```
#include<stdio.h>
nt main(void){
        int data[][2]={{0,1},{2,3}};            //配列宣言と初期化
        int i,j;                                //添え字カウンタ

        printf( “2 次元配列 data¥n” );
        for( i=0;i<2;i++ ){                     //添え字 i のループ
                for( j=0;j<2;j++ ){             //添え字 j のループ
                        printf( “data[%d][%d]=%d “,i,j,data[i][j] );    //配列要素の表示
                }
        printf( “¥n” );
        }
}
```

data[i][j]	j=0	j=1
i=0	0	1
i=1	2	3

図 7 2 次元配列要素の表示プログラム Ⅱ

③　2 次元配列の仕組み

2 次元配列を宣言すると、図 8 に示すようにメモリに各配列要素が格納される

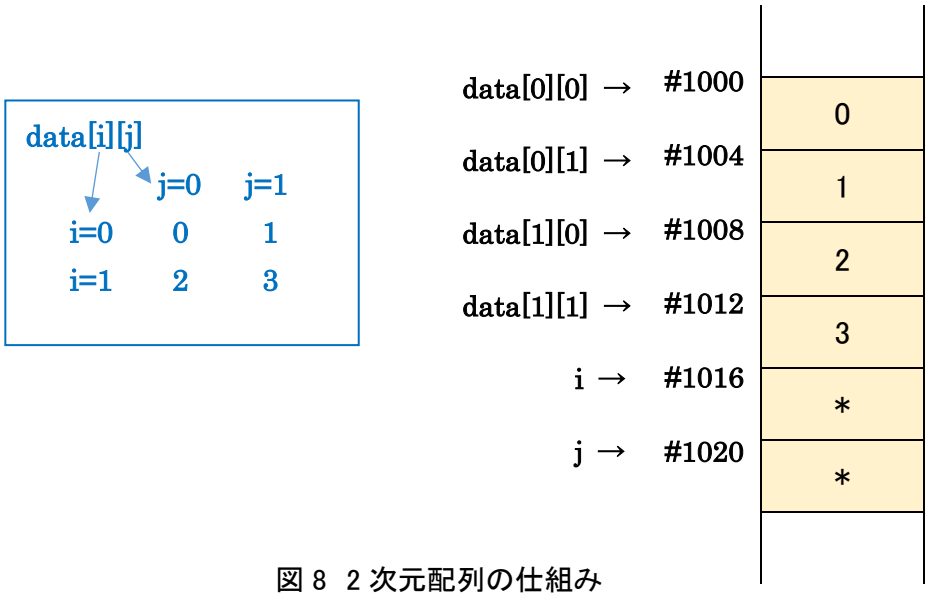

図 8　2 次元配列の仕組み