# 14.2. 配列のメモリ配置

前の節では配列の特徴について解説しました。
ここでは C 言語で配列を使用する場合のメモリ配置について解説します。

## 配列のメモリ配置

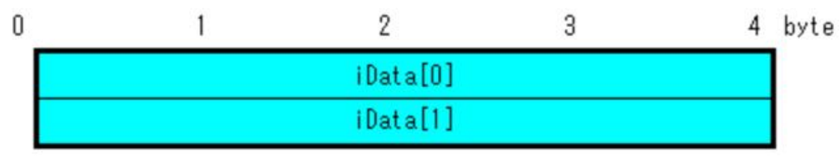
配列は同じデータ型の複数個のデータをまとめて管理するためのメモリ領域です。
データ型の違いにより、メモリ領域の配置も異なります。

### int 型配列のメモリ配置

int 型配列を定義する場合の記述例と定義した配列のメモリ配置を以下に示します。

```
int iData[2];
```

int 型配列の要素数を 2 で定義した場合、メモリ領域上に次のように配置されます。(32 ビット CPU の場合)



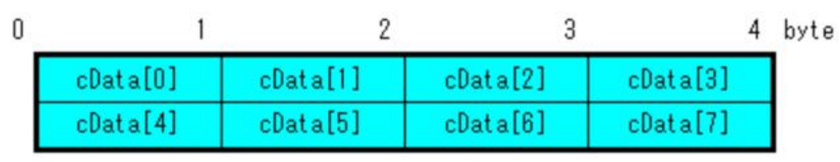

int 型のデータサイズは 4 バイトであるため、メモリ上に連続して 8 バイトの領域が確保されます。

### char 型配列のメモリ配置

char 型配列を定義する場合の記述例と定義した配列のメモリ配置を以下に示します。

```
char cData[8];
```

char 型配列の要素数を 8 で定義した場合、メモリ領域上に次のように配置されます。



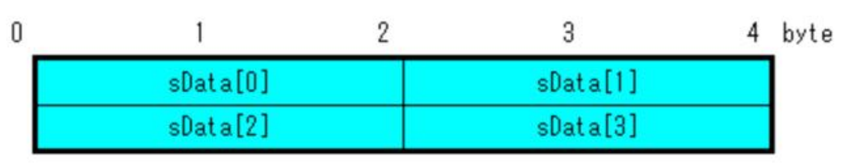

char 型のデータサイズは 1 バイトであるため、メモリ上に連続して 8 バイトの領域が確保されます。

### short 型配列のメモリ配置

short 型配列を定義する場合の記述例と定義した配列のメモリ配置を以下に示します。

```
short sData[4];
```

short 型配列の要素数を 4 で定義した場合、メモリ領域上に次のように配置されます。




short 型のデータサイズは 2 バイトであるため、メモリ上に連続して 8 バイトの領域が確保されます。

## 初期化

配列 cData[8] を宣言し、初期値を設定するサンプルプログラムを以下に示します。

### コード

```
#include <stdio.h>

int main()
{
   char cData[8] = {1, 2, 3};

   printf("cData[0]=%d, cData[1]=%d, cData[2]=%d, cData[3]=%d\n",
         cData[0], cData[1], cData[2], cData[3]);
   printf("cData[4]=%d, cData[5]=%d, cData[6]=%d, cData[7]=%d\n",
         cData[4], cData[5], cData[6], cData[7]);
   return 0;
}
```

### 結果

```
cData[0]=1, cData[1]=2, cData[2]=3, cData[3]=0
cData[4]=0, cData[5]=0, cData[6]=0, cData[7]=0
```

上記プログラム終了後のメモリ配置を以下に示します。


配列の宣言時に設定した初期値は cData[0]、cData[1]、cData[2] に格納されています。

### 配列の初期化

配列の初期化を行う場合、以下のように記述します。

```
char cData[8] = {1, 2, 3};
```

この場合、配列の先頭から初期値が格納されるため、cData[0] ～ cData[2] の初期化が行われます。
初期化を行わない要素は 0 で初期化されます。
※　ただし、配列の宣言時にすべての要素に対して初期化を行っていない場合は、不定値が格納されています。