



配列とポインタの比較

プログラマのためのC言語 第14回

● 概要

- ✓ ポインタと配列は似ている
- ✓ ポインタ演算

“

配列とポインタは似ている

● 配列とポインタは似ている

✓ 配列名 **a** は 配列データの先頭アドレスを表す

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	00	00	00	
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

`short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte`

- 配列とポインタは似ている

✓ ポインタ **p** を用意して配列の先頭アドレスを入れた場合

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	00	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

p

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

```
short* p = a;
```

- 配列とポインタは似ている

✓ ポインタ **p** を用意して配列の先頭アドレスを入れた場合

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	11	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

p

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

```
short* p = a;
```

● 配列とポインタは似ている

✓ ポインタ **p** を用意して配列の先頭アドレスを入れた場合

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	11	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

p

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

```
short* p = a;
```

aは配列のアドレスを表す定数
pは配列のアドレスを入れた変数

“

ポインタ演算

- ポインタ演算

- ✓ ポインタは加減算ができる
- ✓ ポインタはインクリメント、デクリメントできる

- ポインタの加減算

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	00	00	00	
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p

```
short* p = a;  
p += 1;  
p += 2;
```

- ポインタの加減算

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	11	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a p

```
short* p = a;
```

```
p += 1;
```

```
p += 2;
```

● ポインタの加減算

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	13	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a p

```
short* p = a;  
p += 1;  
p += 2;
```

← +1は short型 1個分

● ポインタの加減算

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	17	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p

```
short* p = a;  
p += 1;  
p += 2;
```

← +2は short型 2個分

- ポインタのインクリメント・デクリメント

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	00	00	00	
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p

```
short* p = a;  
p++;  
p++;  
p--;  
p--;
```

- ポインタのインクリメント・デクリメント

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	11	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a p

```
short* p = a;  
p++;  
p++;  
p--;  
p--;
```



- ポインタのインクリメント・デクリメント

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	13	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p

```
short* p = a;  
p++;  
p++;  
p--;  
p--;
```



- ポインタのインクリメント・デクリメント

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	15	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p

```
short* p = a;  
p++;  
p++;  
p--;  
p--;
```

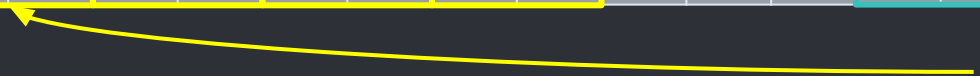
- ポインタのインクリメント・デクリメント

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```


値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	13	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p



```
short* p = a;  
p++;  
p++;  
p--;  
p--;
```



- ポインタのインクリメント・デクリメント

```
short a[] = { 1, 2, 3, 4, 5 }; // 2byte x 5 = 10byte
```

値	00	01	00	02	00	03	00	04	00	05	00	00	00	00	11	00	00	00
番地	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27

a

p

```
short* p = a;  
p++;  
p++;  
p--;  
p--;
```

