「アルゴリズム」資料 2(f)
C の配列とポインタ

鴨浩靖

2009 年 10 月 6 日 初版
2011 年 10 月 4 日 第二版
2012 年 10 月 1 日 第三版
2012 年 10 月 2 日 第三版改
2020 年 10 月 12 日 第四版

配列とポインタ

C 言語では、式の中で配列名が使われると、その配列の最初の要素へのポインタと解釈される。

例

```
int     a[10];
int     *p;
```

と定義されているとき、

```
p = a;
```

と

```
p = &a[0];
```

は同じ意味。

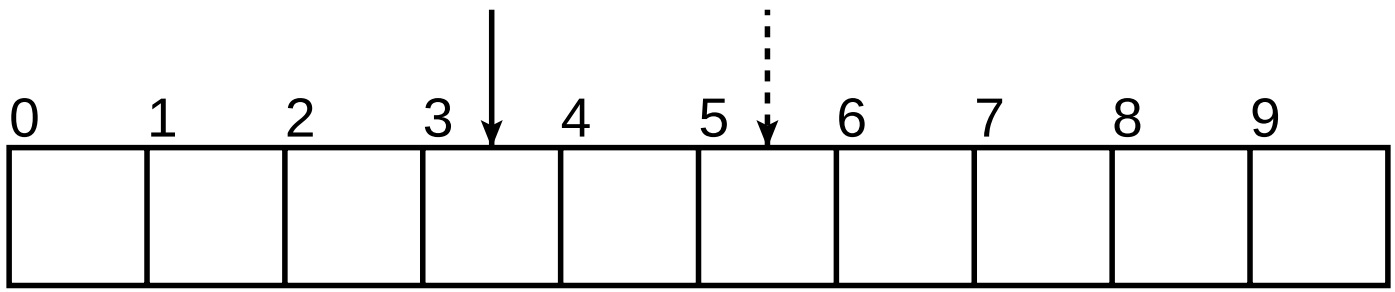
ポインタに整数を足す

配列の要素を指しているポインタ値に整数値を足すと、結果は加えた整数値だけ添字を進めた要素を指すポインタ値になる。

例

```
int     a[10];
int     *p;
```

と定義されていて、p の値が a[3] を指すポインタであるときに、p + 2 を計算すると、結果は a[5] を指すポインタになる。



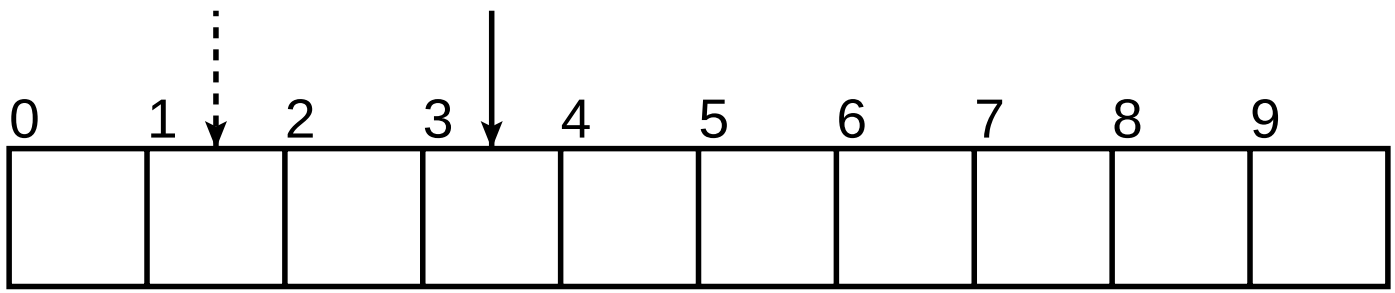
ポインタから整数を引く

配列の要素を指しているポインタ値から整数値を引くと、結果は引いた整数値だけ添字を戻した要素を指すポインタ値になる。

例

```
int     a[10];
int     *p;
```

と定義されていて、p の値が a[3] を指すポインタであるときに、p - 2 を計算すると、結果は a[1] を指すポインタになる。



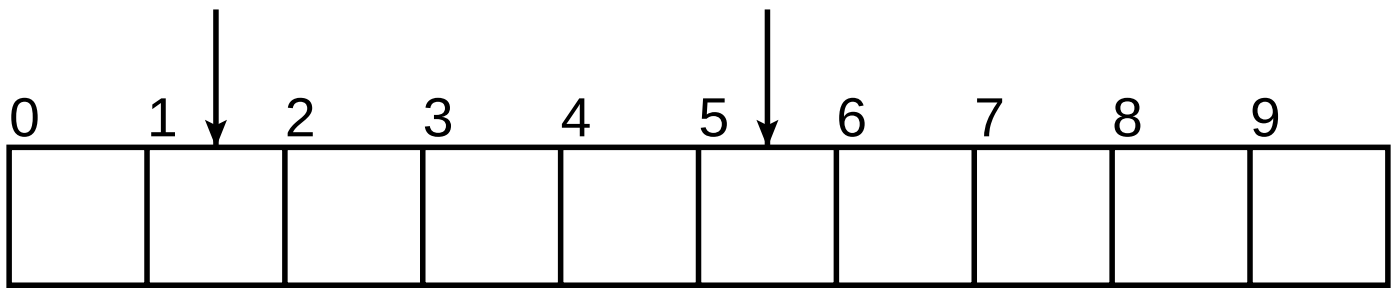
ポインタからポインタを引く

配列の要素を指しているポインタ値から同じ配列の要素を指しているポインタ値を引くと、結果は添字の差にあたる整数値になる。

例

```
int     a[10];
int     *p, *q;
```

と定義されていて、p の値が a[5] を指すポインタであり、q の値が a[1] を指すポインタであるときに、p - q を計算すると結果は 4 になる。q - p を計算すると結果は −4 になる。



ポインタからポインタを引く（つづき）

二つのポインタ値が同じ配列の要素を指すものでなければ、引き算して何が起きるかは処理系に依存する。

例

```
int     a[10];
int     b[20];
int     *p, *q;
```

と定義されていて、p の値が a[5] を指すポインタであり、q の値が b[1] を指すポインタであるときに、p - q を計算して何が起きるかはわからない。

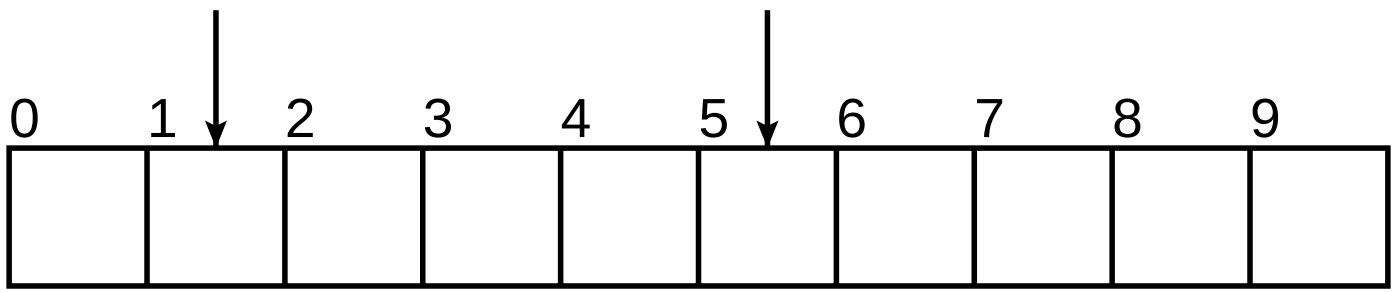
ポインタとポインタの大小比較

配列の要素を指しているポインタ値と同じ配列の要素を指しているポインタ値の大小比較の結果は、添字の大小比較の結果と一致する。

例

```
int     a[10];
int     *p, *q;
```

と定義されていて、p の値が a[5] を指すポインタであり、q の値が a[1] を指すポインタであるときに、p < q を計算すると、結果は 0 になる。p >= q を計算すると、結果は 1 になる。



ポインタとポインタの大小比較（つづき）

二つのポインタ値が同じ配列の要素を指すものでなければ、大小比較で何が起きるかは処理系に依存する。

例

```
int     a[10];
int     b[20];
int     *p, *q;
```

と定義されていて、p の値が a[5] を指すポインタであり、q の値が b[1] を指すポインタであるときに、p < q を計算して何が起きるかはわからない。

配列の要素の総和

a[] の総和を計算して、s に代入したい。

- 添字を動かす

```
int     i;
s = 0;
for (i = 0; i < 100; i ++) {
        s += a[i];
}
```

- ポインタを動かす

```
int     *p;
s = 0;
for (p = &a[0]; p < &a[100]; p ++) {
        s += *p;
}
```

配列の要素の総和（つづき）

以下は、書き方が違うだけで全く同じ意味。

- ポインタを動かす

```
int     *p;
s = 0;
for (p = &a[0]; p < &a[100]; p ++) {
        s += *p;
}
```

- ポインタを動かす

```
int     *p;
s = 0;
for (p = a; p < a + 100; p ++) {
        s += *p;
}
```

配列から配列へのコピー

- 添字を動かす

```
int     i;
for (i = 0; i < 100; i ++) {
        b[i] = a[i];
}
```

- ポインタを動かす

```
int     *p, *q;
for (p = a, q = b; p < a + 100; p ++, q ++) {
        *q = *p;
}
```

配列から配列へのコピー（つづき）

- memcpy() を使う。

  ```
  #include <string.h>
  ```

  が必要。

  ```
  memcpy(b, a, sizeof(int) * 100);
  ```

配列から配列へ各要素を２個ずつコピー

- 添字を動かす

```
int     i;
for (i = 0; i < 100; i ++) {
        b[i * 2] = a[i];
        b[i * 2 + 1] = a[i];
}
```

- 添字を動かす（別解）

```
int     i, j;
for (i = 0, j = 0; i < 100; i ++, j += 2) {
        b[j] = a[i];
        b[j + 1] = a[i];
}
```

配列から配列へ各要素を２個ずつコピー（つづき）

- 添字を動かす（別解２）

```
int     i, j;
for (i = 0, j = 0; i < 100; i ++) {
        b[j] = a[i];
        j ++;
        b[j] = a[i];
        j ++;
}
```

配列から配列へ各要素を２個ずつコピー（つづき 2）

- ポインタを動かす

```
int     *p, *q;
for (p = a, q = b; p < a + 100; p ++, q += 2) {
        *q = *p;
        *(q + 1) = *p;
}
```

配列から配列へ各要素を２個ずつコピー（つづき 3）

- ポインタを動かす（別解）

```
int     *p, *q;
for (p = a, q = b; p < a + 100; p ++) {
        *q = *p;
        q ++;
        *q = *p;
        q ++;
}
```

配列から二つの配列へ交互にコピー

- 添字を動かす

```
int     i;
for (i = 0; i < 50; i ++) {
        b[i] = a[i * 2];
        c[i] = a[i * 2 + 1];
}
```

- 添字を動かす（別解）

```
int     i, j;
for (i = 0, j = 0; j < 50; i += 2, j ++) {
        b[j] = a[i];
        c[j] = a[i + 1];
}
```

配列から二つの配列へ交互にコピー（つづき）

- 添字を動かす（別解 2）

```
int     i, j;
for (i = 0, j = 0; j < 50; j ++) {
        b[j] = a[i];
        i ++;
        c[j] = a[i];
        i ++;
}
```

配列から二つの配列へ交互にコピー（つづき 2）

- ポインタを動かす

```
int     *p, *q, *r;
for (p = a, q = b, r = c; q < b + 50;
     p += 2, q ++, r ++) {
        *q = *p;
        *r = *(p + 1);
}
```

配列から二つの配列へ交互にコピー（つづき 3）

- ポインタを動かす（別解）

```
int     *p, *q, *r;
for (p = a, q = b, r = c; q < b + 50;
     q ++, r ++) {
        *q = *p;
        p ++;
        *r = *p;
        p ++;
}
```

まとめ

今回、学んだこと

- C での配列とポインタの関係
  - ポインタに整数を足す・ポインタから整数を引く
  - ポインタからポインタを引く
  - ポインタとポインタの大小比較
- 応用