# 第6章　メモリ管理

## 6.1　newとdeleteの基本的な使い方

C 言語で動的なメモリを確保する場合、malloc を使います。メモリ開放は free です。

しかし、C++言語では new と delete を使います。C 言語で次のようなコードを書いたとします。

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   int main()
4   {
5           int* p = (int*)malloc( sizeof(int) );
6           if( !p ){
7                   printf("%s\n","memory allocation error.");
8                   return 1;
9           }
10          *p = 10;
11          printf("value : %d",*p);
12          free(p);
13          return 0;
14  }
```

これを new/delete を使って書くと次のようなコードとなります。

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5           int* p = new int;
6           if( !p ) {
```

```
7                   cout << "memory allocation error." << endl;
8                   return 1;
9           }
10          *p = 10;
11          cout << "value : " << *p << endl;
12          delete p;
13          return 0;
14  }
```

new は確保したメモリのポインタを返します。

new の使い方

```
new 型;
```

確保したメモリは delete で削除することができます。

delete の使い方

```
delete 削除する変数;
```

malloc で確保したものを delete で削除や、new で確保したメモリを free で開放等はできません。詳しくは説明しませんが、malloc と new ではメモリを確保する場所が違います。malloc ではヒープ (heap) という場所に確保され、new ではフリーストア (Free Store) という場所に確保されます。

## 6.2 配列のメモリ割り当て

配列のメモリを確保する場合次のようなコードを書きます。

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5   {
6           int* p = new int[3];
7           if( !p ) {
```

```
 8                   cout << "memory allocation error." << endl;
 9                   return 1;
10           }
11           for(int i=0; i < 3; i++ )
12                   p[i] = i;
13
14           for(int i=0; i < 3; i++ )
15                   cout << "value [" << i << "] :" << p[i] << "\n";
16           delete [] p;
17           return 0;
18   }
```

配列の確保

```
new 型 [要素数];
```

要素数 3 個の int 型の配列を確保する場合、例のように `new int[3]` とします。注意してもらいたいのは、delete の方です。

配列の開放

```
delete [] 削除する変数;
```

配列を確保した場合、delete は `delete [] p;` となります。ここで、`delete p;` としてはいけません。こうするとメモリリーク[1]の原因となります。

[1] new したものを delete しない事

# 参考文献

[1] ハーバート・シルト著, トップスタジオ訳, 独習 C++改訂版.