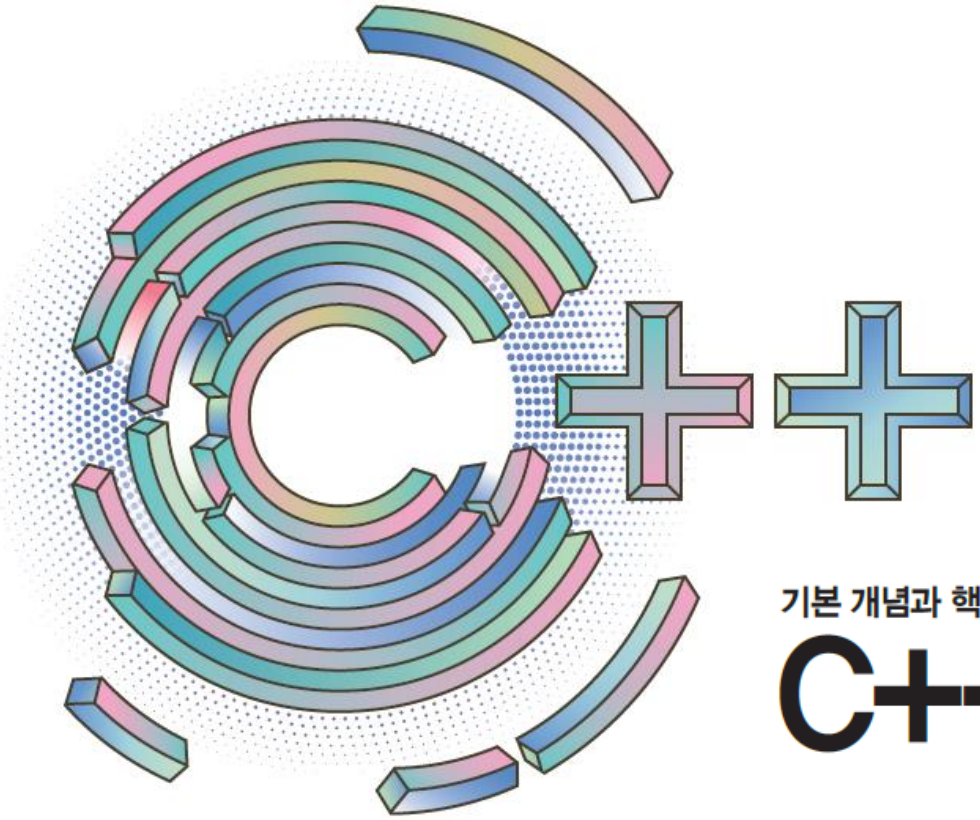
한빛아카데미 Hanbit Academy, Inc.



기본 개념과 핵심 원리로 배우는

C++ 프로그래밍

14장. 메모리 할당과 해제

목차

01. 가상 메모리

■ 가상 메모리 크기

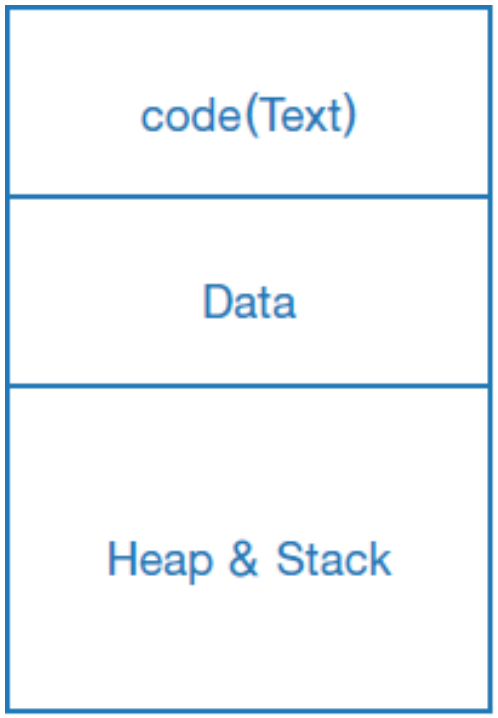
- 가상 메모리는 각각의 프로세스에 독립적으로 부여되는 논리적인 메모리 공간을 나타낸다.
- Windows에서 32비트는 4GB, 64비트는 8TB의 메모리 공간이 부여된다.

■ 가상 메모리 동작 원리

- 프로세스가 시작되면 4GB나 8TB의 메모리 공간이 부여되고, 필요한 메모리를 요청할 경우 해당 메모리 공간에서 페이지 단위로 메모리 영역을 할당해준다

01. 가상 메모리

■ 가상 메모리 구조



[가상 메모리 구조]

01. 가상 메모리

힙

- 힙은 동적으로 메모리를 할당할 때 사용되는 영역이다.

스택

- 스택은 스레드당 하나씩 생성된다.
- 보통 기본적으로 스레드당 1~4MB 정도의 메모리 영역이 스레드당 스택으로 할당된다.

02. 동적 메모리 할당과 해제

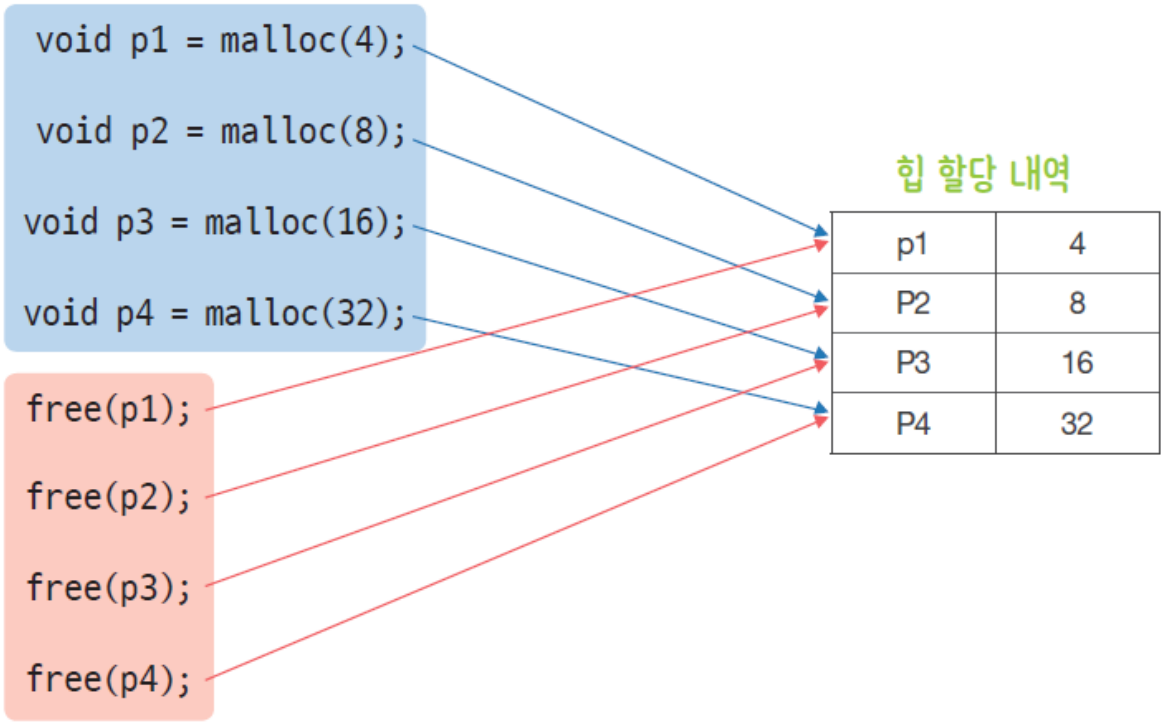
malloc & free

```
void* malloc(size_t size);
void free(void* memblock);
```

```
#ifdef _WIN64
typedef unsigned __int64 size_t;
#else
typedef unsigned int     size_t;
#endif
```

02. 동적 메모리 할당과 해제

malloc & free



[힙 할당 내역]

02. 동적 메모리 할당과 해제

malloc & free

[예제 14-1] malloc, free 사용

```
1  #include <iostream>
2  using namespace std;
3
4  #include <stdlib.h>
5
6  void main()
7  {
8     int a = 1;
9     int* p = (int*)malloc(sizeof(int));
10    *p = 2;
11
```

02. 동적 메모리 할당과 해제

malloc & free

```
12      cout << a << endl;     // 1
13      cout << *p << endl;     // 2
14
15      free(p);
16  }
```

```
1
2
```

02. 동적 메모리 할당과 해제

calloc

[예제 14-2] calloc

```
1  #include <iostream>
2  using namespace std;
3
4  void main()
5  {
6     int* p1 = (int*)malloc(sizeof(int));
7     int* p2 = (int*)calloc(1, sizeof(int));
8
9     cout << *p1 << endl;
10    cout << *p2 << endl;
11
12    free(p1);
13    free(p2);
14 }
```

```
4390988
0
```

02. 동적 메모리 할당과 해제

realloc

```
void* realloc(void* memblock, size_t size);
```

[예제 14-3] calloc

```
1  #include <iostream>
2  using namespace std;
3
4  void main()
5  {
6     int* p1 = (int*)malloc(sizeof(int));
7     *p1 = 3;
8
9     int* p2 = (int*)realloc(p1, 1024 * sizeof(int));
```

02. 동적 메모리 할당과 해제

realloc

```
10
11      cout << p1 << endl;
12      cout << p2 << endl;
13      cout << p2[0] << endl;
14
15      free(p1);    // Error
16      free(p2);    // OK
17  }
```

```
007BB128
007BDD60
3
```

02. 동적 메모리 할당과 해제

new & delete

```
TYPE* pT = new TYPE;
delete pT;
```

[예제 14-4] malloc

```
1  #include <iostream>
2  using namespace std;
3
4  void main()
5  {
6     int* p1 = (int*)malloc(sizeof(int));
7     *p1 = 3;
8     cout << *p1 << endl;   // 3
9     free(p1);
10
```

02. 동적 메모리 할당과 해제

new & delete

```
11      int* p2 = new int;
12      *p2 = 3;
13      cout << *p2 << endl;    // 3
14      delete p2;
15  }
```

```
3
3
```

02. 동적 메모리 할당과 해제

new [] & delete []

```
TYPE* pT = new TYPE[N];
delete [] pT;
```

02. 동적 메모리 할당과 해제

new [] & delete []

[예제 14-5] new [] & delete []

```
1  void main()
2  {
3     int* p1 = (int*)malloc(2 * sizeof(int));
4     p1[0] = 0;
5     p1[1] = 1;
6     free(p1);
7
8     int* p2 = new int[2];
9     p2[0] = 0;
10    p2[1] = 1;
11    delete [] p2;
12 }
```

02. 동적 메모리 할당과 해제

malloc과 new의 차이점

[예제 14-6] malloc vs new

```
1  #include <iostream>
2  using namespace std;
3
4  class CTest
5  {
6  public:
7     CTest()
8     {
9        cout << "Constructor!" << endl;
10    }
11 };
12
13 void main()
```

02. 동적 메모리 할당과 해제

malloc과 new의 차이점

```
14  {
15      cout << "<malloc>" << endl;
16      CTest* pT1 = (CTest*)malloc(sizeof(CTest));
17      free(pT1);
18
19      cout << "<new>" << endl;
20      CTest* pT2 = new CTest;
21      delete pT2;
22  }
```

```
<malloc>
<new>
Constructor!
```

02. 동적 메모리 할당과 해제

malloc과 new의 차이점

[예제 14-7] malloc vs new []

```
1  #include <iostream>
2  using namespace std;
3
4  class CTest
5  {
6  public:
7     CTest()
8     {
9        cout << "Constructor!" << endl;
10    }
11 };
12
13 void main()
```

02. 동적 메모리 할당과 해제

malloc과 new의 차이점

```
14  {
15      cout << "<malloc array>" << endl;
16      CTest* pT1 = (CTest*)malloc(2 * sizeof(CTest));
17      free(pT1);
18
19      cout << "<new []>" << endl;
20      CTest* pT2 = new CTest[2];
21      delete [] pT2;
22  }
```

```
<malloc array>
<new []>
Constructor!
Constructor!
```

02. 동적 메모리 할당과 해제

■ free와 delete의 차이점

[예제 14-8] free, delete, delete []

```
1  #include <iostream>
2  using namespace std;
3
4  class CTest
5  {
6  public:
7     ~CTest()
8     {
9        cout << "Destructor!" << endl;
10    }
11 };
12
13 void main()
```

02. 동적 메모리 할당과 해제

■ free와 delete의 차이점

```
14  {
15      cout << "<free>" << endl;
16      CTest* pT1 = (CTest*)malloc(sizeof(CTest));
17      free(pT1);
18
19      cout << "<delete>" << endl;
20      CTest* pT2 = new CTest;
21      delete pT2;
22
23      cout << "<delete []>" << endl;
24      CTest* pT3 = new CTest[2];
25      delete [] pT3;
26
27      cout << "<delete void*>" << endl;
28      CTest* pT4 = new CTest;
```

02. 동적 메모리 할당과 해제

■ free와 delete의 차이점

```
29      delete (void*)pT4;
30  }
```

```
<free>
<delete>
Destructor!
<delete []>
Destructor!
Destructor!
<delete void*>
```

02. 동적 메모리 할당과 해제

delete와 delete [ ]

[예제 14-9] free, delete, delete []

```
1  #include <iostream>
2  using namespace std;
3
4  class CTest
5  {
6  public:
7     CTest()
8     {
9        cout << "Constructor!" << endl;
10    }
11
12    ~CTest()
13    {
```

02. 동적 메모리 할당과 해제

delete와 delete [ ]

```
14          cout << "Destructor!" << endl;
15      }
16
17      int m_Val1;
18      int m_Val2;
19  };
20
21  CTest* g_pT = NULL;
22
23  void FuncNew()
24  {
25      cout << "<FuncNew>" << endl;
26      g_pT = new CTest[2];
27  }
28
```

02. 동적 메모리 할당과 해제

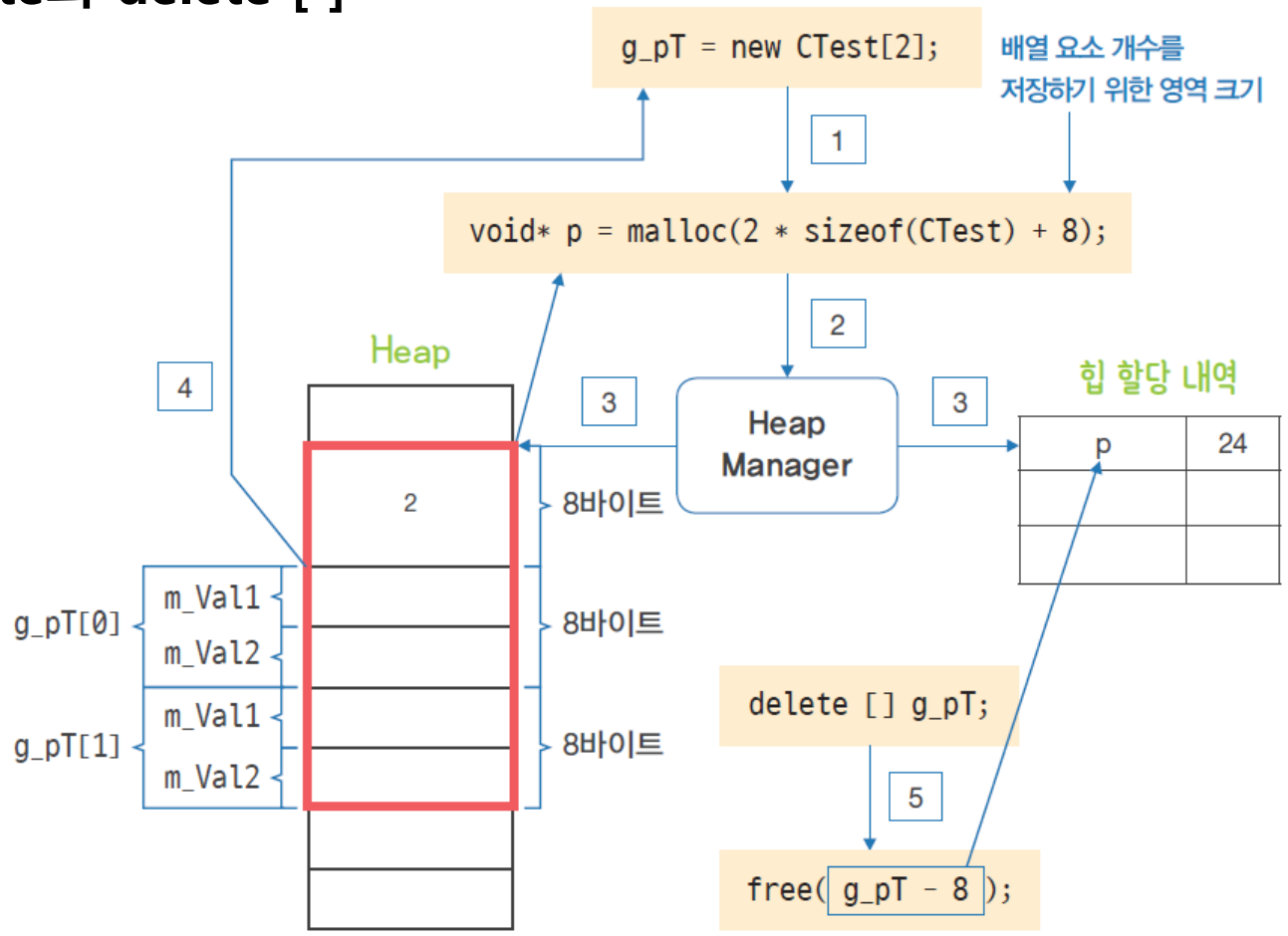
delete와 delete [ ]

```
29  void FuncDelete()
30  {
31      cout << "<FuncDelete>" << endl;
32      delete [] g_pT;
33  }
34
35  void main()
36  {
37      FuncNew();
38      FuncDelete();
39  }
```

```
<FuncNew>
Constructor!
Constructor!
<FuncDelete>
Destructor!
Destructor!
```

02. 동적 메모리 할당과 해제

delete와 delete [ ]



[new [ ]의 힙 할당 내역]

03. 메모리 할당과 해제 연산자 함수

operator new

[예제 14-10] 기본 operator new

```
1  void* operator new(size_t const size)
2  {
3     if(void* const block = malloc(size))
4     {
5        return block;
6     }
7
8     // 예외 처리
9  }
```

03. 메모리 할당과 해제 연산자 함수

operator new

[예제 14-11] operator new 중복 정의 1

```
1  #include <iostream>
2  using namespace std;
3
4  void* operator new(size_t size)
5  {
6     void* p = malloc(size);
7     memset(p, 0xFF, size);
8     return p;
9  }
10
11 void main()
12 {
13    int* p = new int;
```

03. 메모리 할당과 해제 연산자 함수

operator new

```
14      cout << *p << endl;     // -1 출력
15      delete p;
16  }
```

```
-1
```

03. 메모리 할당과 해제 연산자 함수

operator new

[예제 14-12] operator new 중복 정의 2

```
1  #include <iostream>
2  using namespace std;
3
4  void* operator new(size_t size, int value)
5  {
6     void* p = malloc(size);
7     memset(p, value, size);
8     return p;
9  }
10
11 void main()
12 {
13    int* p1 = new(0) int;
```

03. 메모리 할당과 해제 연산자 함수

operator new

```
14      cout << *p1 << endl;    // 0 출력
15      delete p1;
16
17      int* p2 = new(-1) int;
18      cout << *p2 << endl;    // -1 출력
19      delete p2;
20  }
```

```
0
-1
```

03. 메모리 할당과 해제 연산자 함수

operator new

[예제 14-13] operator new 중복 정의 3

```
1  #include <iostream>
2  using namespace std;
3
4  size_t g_Index = 0;
5  char g_Data[1024] = {0};
6
7  void* operator new(size_t size)
8  {
9     void* p = &g_Data[g_Index];
10    g_Index += size;
11    return p;
12 }
13
```

03. 메모리 할당과 해제 연산자 함수

operator new

```
14  void main()
15  {
16      char* p1 = new char;
17      char* p2 = new char;
18      char* p3 = new char;
19
20      *p1 = 'C';
21      *p2 = '+';
22      *p3 = '+';
23
24      cout << g_Data << endl;     // C++ 출력
25  }
```

```
C++
```

03. 메모리 할당과 해제 연산자 함수

operator new

```
(static) void* CLASS_NAME::operator new(size_t const size);
```

03. 메모리 할당과 해제 연산자 함수

operator new

[예제 14-14] 멤버 함수 operator new 중복 정의

```
1  #include <iostream>
2  using namespace std;
3
4  class CTest
5  {
6  public:
7     static void* operator new(size_t size)
8     {
9        void* p = malloc(size);
10       *(int*)p = 7;
11       cout << "CTest::operator new" << endl;
12       return p;
13    }
```

03. 메모리 할당과 해제 연산자 함수

operator new

```
14
15      int m_Value;
16  };
17
18  void main()
19  {
20      CTest* p = new CTest;
21      cout << p->m_Value << endl;     // 7 출력
22      delete p;
23  }
```

```
CTest::operator new
7
```

03. 메모리 할당과 해제 연산자 함수

operator delete

[예제 14-15] 기본 operator delete

```
1  void operator delete(void* const block)
2  {
3     free(block);
4  }
```

03. 메모리 할당과 해제 연산자 함수

operator new [ ] & delete [ ]

[예제 14-16] 기본 operator new [] & operator delete []

```
1  void* operator new[](size_t const size)
2  {
3     return operator new(size);
4  }
5
6  void operator delete[](void* block)
7  {
8     operator delete(block);
9  }
```

Thank You !

한빛아카데미 Hanbit Academy, Inc.