자료구조 HW1

MyIntVector

2019203102 유지성

실행 환경

Microsoft Visual Studio Community 2019
버전 16.6.0
VisualStudio.16.Release/16.6.0+30114.105
Microsoft .NET Framework
버전 4.8.03752

설치된 버전: Community

Visual C++ 2019 00435-60000-00000-AA308
Microsoft Visual C++ 2019

ASP.NET and Web Tools 2019 16.6.936.3669
ASP.NET and Web Tools 2019

C# 도구 3.6.0-4.20251.5+910223b64f108fcf039012e0849befb46ace6e66
IDE에서 사용되는 C# 구성 요소입니다. 프로젝트 형식 및 설정에 따라 다른 버전의 컴파일러를 사용할 수 있습니다.

IntelliCode 확장 1.0
IntelliCode Visual Studio 확장 세부 정보

Microsoft JVM Debugger 1.0
Provides support for connecting the Visual Studio debugger to JDWP compatible Java Virtual Machines

Microsoft MI-Based Debugger 1.0
Provides support for connecting Visual Studio to MI compatible debuggers

Microsoft Visual C++ Wizards 1.0
Microsoft Visual C++ Wizards

Microsoft Visual Studio VC 패키지 1.0

Microsoft Visual Studio VC 패키지

NuGet 패키지 관리자 5.6.0
Visual Studio의 NuGet 패키지 관리자입니다. NuGet에 대한 자세한 내용은 https://docs.nuget.org/를 참조하세요.

ProjectServicesPackage Extension 1.0
ProjectServicesPackage Visual Studio Extension Detailed Info

Test Adapter for Boost.Test 1.0
Enables Visual Studio's testing tools with unit tests written for Boost.Test. The use terms and Third Party Notices are available in the extension installation directory.

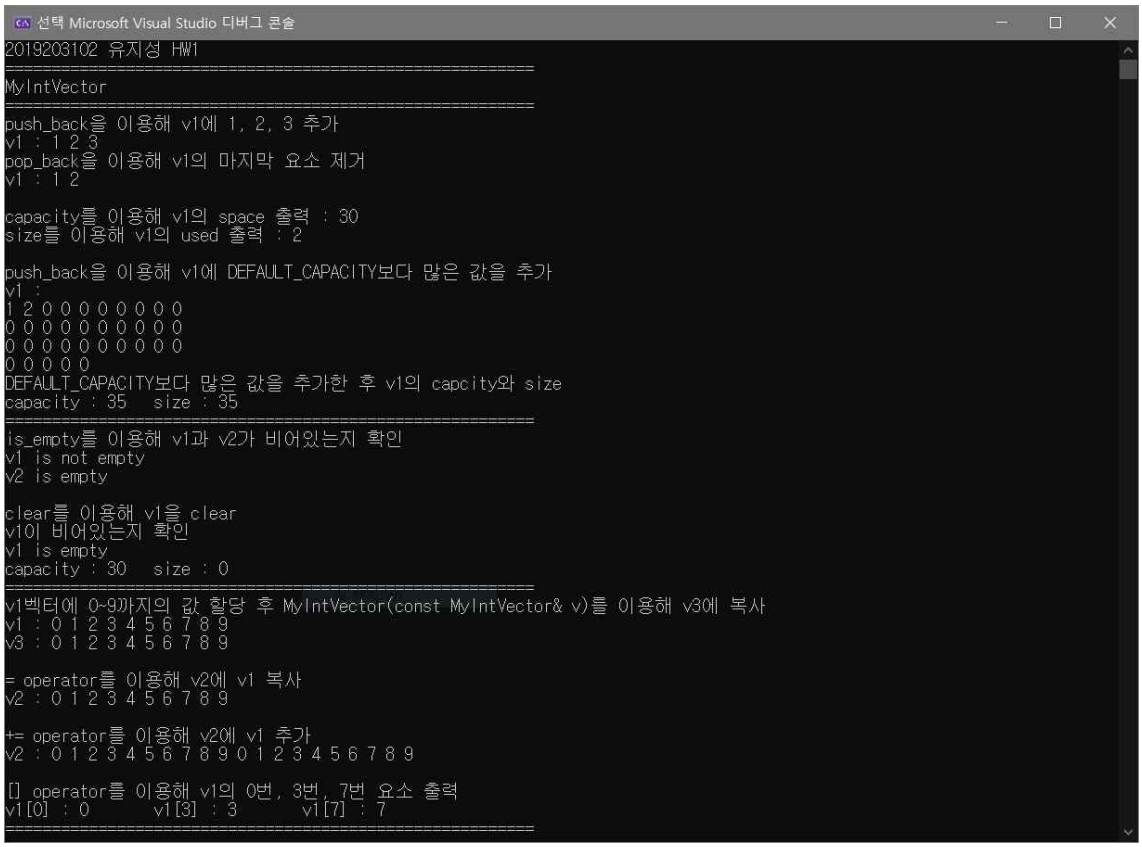
Test Adapter for Google Test 1.0
Google Test용으로 작성된 단위 테스트와 함께 Visual Studio의 테스트 도구를 사용합니다. 사용 약관 및 타사 고지 사항은 확장 설치 디렉터리에서 확인할 수 있습니다.

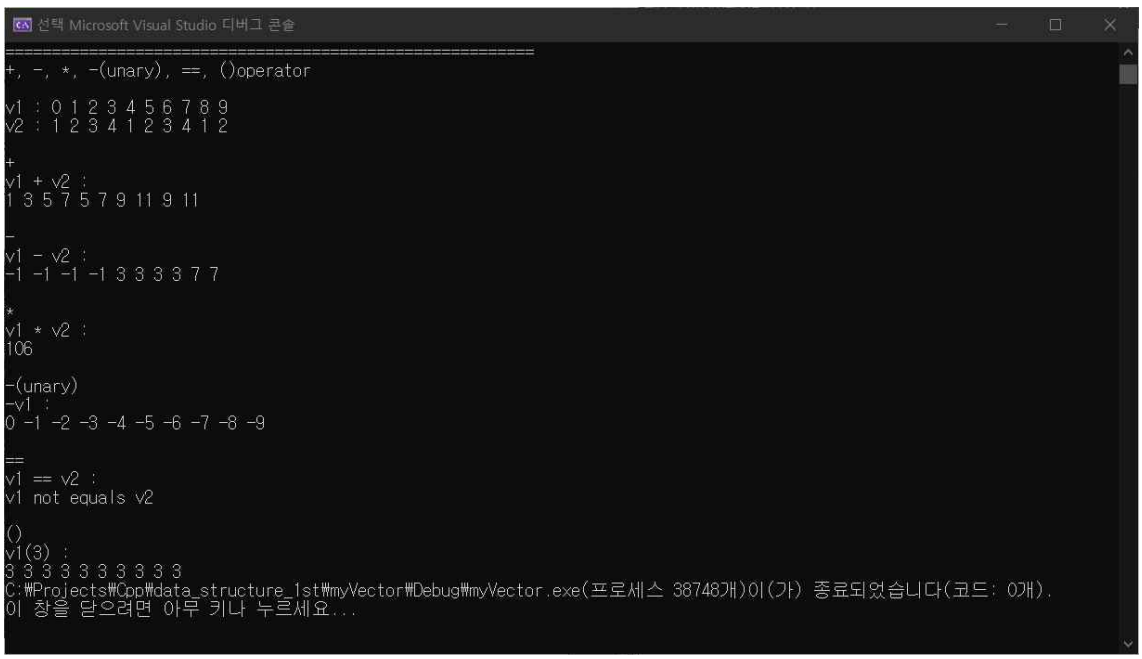
Visual Basic 도구
3.6.0-4.20251.5+910223b64f108fcf039012e0849befb46ace6e66
IDE에서 사용되는 Visual Basic 구성 요소입니다. 프로젝트 형식 및 설정에 따라 다른 버전의 컴파일러를 사용할 수 있습니다.

Visual Studio Code 디버그 어댑터 호스트 패키지 1.0
Visual Studio에서 Visual Studio Code 디버그 어댑터를 호스트하기 위한 Interop 계층

Visual Studio Tools for CMake 1.0
Visual Studio Tools for CMake

실행 예제


선택 Microsoft Visual Studio 디버그 콘솔
2019203102 유지성 HW1
MyIntVector
push_back을 이용해 v1에 1, 2, 3 추가
v1 : 1 2 3
pop_back을 이용해 v1의 마지막 요소 제거
v1 : 1 2
capacity를 이용해 v1의 space 출력 : 30
size를 이용해 v1의 used 출력 : 2
push_back을 이용해 v1에 DEFAULT_CAPACITY보다 많은 값을 추가
v1 :
1 2 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0
DEFAULT_CAPACITY보다 많은 값을 추가한 후 v1의 capcity와 size
capacity : 35 size : 35
is_empty를 이용해 v1과 v2가 비어있는지 확인
v1 is not empty
v2 is empty
clear를 이용해 v1을 clear
v1이 비어있는지 확인
v1 is empty
capacity : 30 size : 0
v1벡터에 0~9까지의 값 할당 후 MyIntVector(const MyIntVector& v)를 이용해 v3에 복사
v1 : 0 1 2 3 4 5 6 7 8 9
v3 : 0 1 2 3 4 5 6 7 8 9
= operator를 이용해 v2에 v1 복사
v2 : 0 1 2 3 4 5 6 7 8 9
+= operator를 이용해 v2에 v1 추가
v2 : 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
[] operator를 이용해 v1의 0번, 3번, 7번 요소 출력
v1[0] : 0 v1[3] : 3 v1[7] : 7


선택 Microsoft Visual Studio 디버그 콘솔
+, -, *, -(unary), ==, ()operator
v1 : 0 1 2 3 4 5 6 7 8 9
v2 : 1 2 3 4 1 2 3 4 1 2
+
v1 + v2 :
1 3 5 7 5 7 9 11 9 11
-
v1 - v2 :
-1 -1 -1 -1 3 3 3 3 7 7
*
v1 * v2 :
106
-(unary)
-v1 :
0 -1 -2 -3 -4 -5 -6 -7 -8 -9
==
v1 == v2 :
v1 not equals v2
()
v1(3) :
3 3 3 3 3 3 3 3 3 3
C:₩Projects₩Cpp₩data_structure_1st₩myVector₩Debug₩myVector.exe(프로세스 38748개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...

```
12      MyIntVector v1;
13      MyIntVector v2;
```

MyIntVector v1과 v2 선언

```
21      cout << "push_back을 이용해 v1에 1, 2, 3 추가₩n";
22      v1.push_back(1);
23      v1.push_back(2);
24      v1.push_back(3);
25      cout << "v1 : ";
26      for (int i = 0; i < v1.size(); i++)
27          cout << v1[i] << " ";
```

push_back으로 요소 추가

```
29      cout << "₩npop_back을 이용해 v1의 마지막 요소 제거₩n";
30      v1.pop_back();
31      cout << "v1 : ";
32      for (int i = 0; i < v1.size(); i++)
33          cout << v1[i] << " ";
```

pop_back으로 요소 제거

```
35      cout << "₩n₩ncapacity를 이용해 v1의 space 출력 : " << v1.capacity() << "₩n";
36      cout << "size를 이용해 v1의 used 출력 : " << v1.size() << "₩n";
```

capacity와 size 출력

```
38      cout << "₩npush_back을 이용해 v1에 DEFAULT_CAPACITY보다 많은 값을
39      cout << "v1 : ₩n";
40      for (int i = 0; i < 33; i++)
41      {
42          v1.push_back(0);
43      }
44      for (int i = 0; i < v1.size(); i++)
45      {
46          cout << v1[i] << " ";
47          if ((i + 1) % 10 == 0 )
48              cout << "₩n";
49      }
```

DEFAULT_CAPACITY보다 많은 값을 추가해서 reserve 함수 사용

```
55      cout << "₩nis_empty를 이용해 v1과 v2가 비어있는지 확인₩n";
56      if (v1.is_empty())
57          cout << "v1 is empty₩n";
58      else
59          cout << "v1 is not empty₩n";
60      if (v2.is_empty())
61          cout << "v2 is empty₩n";
62      else
63          cout << "v2 is not empty₩n";
```

is_empty 사용

```
65      cout << "₩nclear를 이용해 v1을 clear₩n";
66      v1.clear();
67      cout << "v1이 비어있는지 확인₩n";
68      if (v1.is_empty())
69          cout << "v1 is empty₩n";
70      else
71          cout << "v1 is not empty₩n";
```

clear 사용

```
76      cout << "₩nv1벡터에 0~9까지의 값 할당 후 MyIntVector(const MyInt
77      for (int i = 0; i < 10; i++)
78          v1.push_back(i);
79      MyIntVector v3(v1);
```

MyIntVector(const MyIntVector& v)를 이용해 복사와 동시에 MyIntVector 선언

```
87      cout << "₩n₩n= operator를 이용해 v2에 v1 복사₩n";
88      v2 = v1;
89      cout << "v2 : ";
90      for (int i = 0; i < v2.size(); i++)
91          cout << v2[i] << " ";
```

=을 사용해 복사

```
93      cout << "₩n₩n+= operator를 이용해 v2에 v1 추가₩n";
94      v2 += v1;
95      cout << "v2 : ";
96      for (int i = 0; i < v2.size(); i++)
97          cout << v2[i] << " ";
```

+=을 사용해 배열 추가

```
99      cout << "₩n₩n[] operator를 이용해 v1의 0번, 3번, 7번 요소 출력₩n";
100     cout << "v1[0] : " << v1[0] << "₩tv1[3] : " << v1[3] << "₩tv1[7] : " << v1[7] << "₩n";
```

[]로 각 요소 출력

```
114     cout << "₩n₩n+₩n";
115     cout << "v1 + v2 : ₩n";
116     v3 = v1 + v2;
117     for (int i = 0; i < v3.size(); i++)
118         cout << v3[i] << " ";
```

+ 사용

```
120     cout << "₩n₩n-₩n";
121     cout << "v1 - v2 : ₩n";
122     v3 = v1 - v2;
123     for (int i = 0; i < v3.size(); i++)
124         cout << v3[i] << " ";
```

- 사용

```
126     cout << "₩n₩n*₩n";
127     cout << "v1 * v2 : ₩n";
128     int t = v1 * v2;
129     cout << t;
```

* 사용

```
131     cout << "₩n₩n-(unary)₩n";
132     cout << "-v1 : ₩n";
133     v3 = -v1;
134     for (int i = 0; i < v3.size(); i++)
135         cout << v3[i] << " ";
```

-(unary) 사용

```
137     cout << "₩n₩n==₩n";
138     cout << "v1 == v2 : ₩n";
139     if (v1 == v2)
140         cout << "v1 equals v2";
141     else
142         cout << "v1 not equals v2";
```

== 사용

```
144     cout << "₩n₩n()₩n";
145     cout << "v1(3) : ₩n";
146     v3 = v1(3);
147     for (int i = 0; i < v3.size(); i++)
148         cout << v3[i] << " ";
```

() 사용

예외 처리

reserve

```
51  void MyIntVector::reserve(size_t n)
52  {
53      if (n < 0)
54          throw "Error! (reserve) : capacity(space) should not lower than 0";
```

n이 음수일 경우 예외 처리

operator []

```
112 int MyIntVector::operator[](int n)
113 {
114     if (n < used && 0 <= n)
115         return this->data[n];
116     else
117     {
118         cout << "₩nError! ([] operator) : please check range₩n";
119         exit(-1);
120     }
121 }
```

n이 배열의 인덱스 값을 초과할 경우 예외 처리 (프로그램 바로 종료)

operator +, operator -, operator *

```
123 MyIntVector MyIntVector::operator+(const MyIntVector& source)
124 {
125     if (this->used != source.used)
126     {
127         throw "Error! (+ operator) : size are different!₩n";
128     }
```

연산자의 좌항과 우항의 배열 size가 다를 경우 예외 처리

고찰

예외 처리는 throw, try, catch로 처리하였다. 다만, operator []는 범위를 벗어날 시 exit로 프로그램을 바로 종료하도록 하였다.

push_back으로 capacity보다 많은 값을 추가하려 할 때 reserve 함수를 호출해 capacity를 하나씩 늘림으로써 메모리 공간을 효율적으로 활용하도록 하였다.

pop_back 함수 사용시에 pop_back된 인덱스의 요소 값을 0으로 만들도록 하였다.

clear 함수 사용시 used 값을 0으로 설정하고 DEFAULT_CAPACITY로 reserve를 하도록 하여 새로운 공간을 배정받도록 하였다.

+, -, -(unary), () operator에서는 새로운 MyIntVector를 만들어 새 MyIntVector에 계산 결과를 저장한 후 새 MyIntVector를 return하도록 하였다.

예외 처리를 어떻게 할지 고민을 많이 했다. 처음에는 assert로 처리 하려 했으나, assert가 release 모드에서는 사용 할 수 없다고 해서 if else 문으로 처리하려 했다. 그러나 if else 문으로 처리하면 void 형이 아닌 함수에서 안전하게 빠져나오기가 어려웠다. 예를 들어 MyIntVector형인 함수를 보자. return 으로 빠져나올려면 MyIntVector를 return 하여야만 하는데, 임시로 return한 MyIntVector가 예상치 못한 버그를 만들어 낼 수 있다. 그리하여

throw try catch 문으로 예외 처리를 하도록 하였다. throw로 예외처리를 하려면 try catch문을 따로 추가하여야만 하지만, return 타입에 상관 없이 함수를 빠져 나올 수 있어 안전한 예외 처리가 가능했다.