11-2 實作 Vec 的細節

一、建構與解構函數

在建構函數中，我們要初始化 Vec 的 data member。因為剛建立好的 Vec 會是一個空的容器，所以一開始 m_size 和 m_capacity 都是 0。而 m_data 則先賦予它 nullptr 值，表示目前沒有指向任何記憶體。

```
Vec::Vec()
{
    m_data = nullptr;
    m_size = 0;
    m_capacity = 0;
}
```

在解構函數部分，我們先檢查是不有向作業系統要求配置記憶體，若有的話 m_data 將不會是 nullptr，而是會指向那塊記憶體。我們要釋放配置的記憶，把它還給作業系統。

```
Vec::~Vec()
{
    if(m_data!=nullptr)
    {
        delete [] m_data;
    }
}
```

二、size 和 capacity

Vec 的 size 和 capacity 會隨著 push_back 放入愈來愈多的元素而成長，這兩個 data member 不應該被使用者修改，所以我們只提供兩個相應的成員函數用來回傳其值。

```
int Vec::size()
{
    return m_size;
}
```

```
int Vec::capacity()
{
    return m_capacity;
}
```

三、push_back

在 Vec 空間需要成長時，我們採用一個簡單的策略 - 每次成長為之前的 2倍大。

要留意的是，由於一開始 capacity 是 0，乘以 2 之後還是 0，所以要特別處理這個狀況，在第一次 push_back 時，capacity 要成長為 1。

在成長的部分，流程如下。

1. 配置一塊原來 2 倍大的記憶體空間
2. 把舊資料搬到新空間
3. 釋放舊空間記憶體，還給作業系統
4. 將 m_data 指向新的這塊記憶體

```
void Vec::push_back(int val)
{
    // 如果空間不夠用
    if(m_size==m_capacity)
    {
        // 每次成長為原來的 2 倍大小
```

```
        int new_capacity = m_capacity*2;

        if(new_capacity==0)
            new_capacity = 1;

        // 配置一塊新的空間
        int *new_data = new int[new_capacity];
        // 把資料搬到新的空間
        for(int i=0; i<m_size; i++)
        {
            new_data[i] = m_data[i];
        }
        // 釋放舊的空間，改用新空間
        if(m_data!=nullptr)
            delete [] m_data;

        m_data = new_data;
        m_capacity = new_capacity;
    }

    m_data[m_size] = val;
    m_size++;
}
```

四、pop_back()

pop_back 部分很簡單，只要把 size - 1 即可，不必清除那塊記憶體。下次 push_back 時，新資料就會蓋掉它。

```
void Vec::pop_back()
{
    m_size--;
}
```

Revision #1
★Created 8 June 2026 00:12:59 by huihui
Updated 8 June 2026 00:41:43 by huihui