C++

Hoorcollege 2

Bas Terwijn en Robin Wacanno

Programmeertalen

my_vector

Variabelen

```
class my_vector
{
    int m_capacity;
    int m_size;
    T* m_data;

public:
    using value_type = T;
    ...
```

Constructor

```
my_vector(int size=0)
    : m_data{nullptr}, m_capacity{size}, m_size{size}
{
    if (size>0)
    {
        m_data = new T[m_capacity];
        std::fill(m_data, m_data+m_capacity, 0);
    }
}
```

Destructor

```
virtual ~my_vector()
{
    if (m_data!=nullptr)
        delete[] m_data;
}
```

Copy constructor

```
my_vector(const my_vector<T>& v2)
      : m_data{nullptr}, m_capacity{v2.m_size}, m_size{v2.m_size}
{
    if (m_size>0)
    {
        m_data=new T[m_size];
        std::copy(v2.m_data,v2.m_data+m_size,m_data);
    }
}
```

Assignment-operator

```
my_vector& operator=(const my_vector<T>& v2)
{
    if (this!=&v2)
    {
        if (m_data!=nullptr)
            delete[] m_data;
        m_capacity=v2.m_size;
        m_size=v2.m_size;
        m_data=new T[m_size];
        std::copy(v2.m_data,v2.m_data+m_size,m_data);
    }
    return *this;
}
```

Push back

```
void push_back(T t)
{
    if (m_size==m_capacity)
    {
        m_capacity= (m_capacity==0 ? 1 : m_capacity*2);
        T* new_data= new T[m_capacity];
        std::copy(m_data,m_data+m_size,new_data);
        delete[] m_data;
        m_data=new_data;
    }
    m_data[m_size++]=t;
}
```

Index-operator

```
T& operator[](int i) {
    return m_data[i];
}

const T& operator[](int i) const {
    return m_data[i];
}
```

Iterator - methodes

```
iterator begin() {
    return iterator(m_data);
}

iterator end() {
    return iterator(m_data, size());
}
```

Iterator - klasse

```
class iterator {
    int m_i;
    T* m_data;
public:
    iterator() {}
    iterator(T* data,int i=0) : m_data{data},m_i{i} {}
    ...
};
```

Iterator - klasse

```
class iterator {
    ...
    iterator& operator++() {
        ++m_i; return *this;
    }
    iterator operator++(int) {
        iterator retval=*this;
        ++(*this); return retval;
    }
    ...
};
```

Iterator - klasse

```
class iterator {
    ...
    iterator& operator+=(int i) {
        m_i+=i;
        return *this;
    }
    iterator operator+(int i) {
        iterator retval=*this;
        retval+=i;
        return retval;
    }
    ...
};
```

Iterator - klasse

```
class iterator {
    ...
    bool operator==(iterator other) const {
        return m_i==other.m_i;
    }
    bool operator!=(iterator other) const {
        return !(*this==other);
    }
    T& operator*() {
        return m_data[m_i];
    }
};
```

Veelvoorkomende fouten

malloc vs. **new**

```
m_data = (T*)malloc(sizeof(...));

m_data = new T[m_capacity];
```

size = 0

```
my_vector(unsigned int size=0) {
    m_data = new T[size];
    m_capacity = size;
    m_size = 0;
    ...
```

Self assignment

```
my_vector& operator=(const my_vector<T>& v2) // assignment operator, ca
{
    if (this!=&v2)
    {
        if (m_data!=nullptr)
            delete[] m_data; // deallocates old memory
        m_capacity=v2.m_size;
        m_size=v2.m_size;
        m_data=new T[m_size]; // allocates new memory
        std::copy(v2.m_data,v2.m_data+m_size,m_data);
    }
    return *this;
}
```

C vs. C++

C vs. C++

- `malloc(...); free(...);`

  vs.

  `new ...; delete ...;`
- `template<typename T>`

  vs.

  `void* ...`