MOVE СЕМАНТИКИ

доц. д-р Нора Ангелова

MOVE КОНСТРУКТОР

- Подаване на стойност по референция

```
void printValue(int & value) {
  //Some magical code...
}

int main() {
  int val = 10;     // Заделяне на памет
  printValue(val); // Предаване на фактически параметър
  printValue(5);   // Грешка при компилация
  printValue(val+3); // Грешка при компилация

  int& newValName = val; // Коректно присвояване
  int& newFiveName = 5;  // Грешка при компилация

  return 0;
}
```

MOVE КОНСТРУКТОР

- Подаване на стойност по референция

```
void printValue(int const & value) {
  //Some magical code...
}

int main() {
  int val = 10;     // Заделяне на памет
  printValue(val); // Предаване на фактически параметър
  printValue(5);   // Валидно извикване
  printValue(val+3); // Валидно извикване

return 0;
}
```

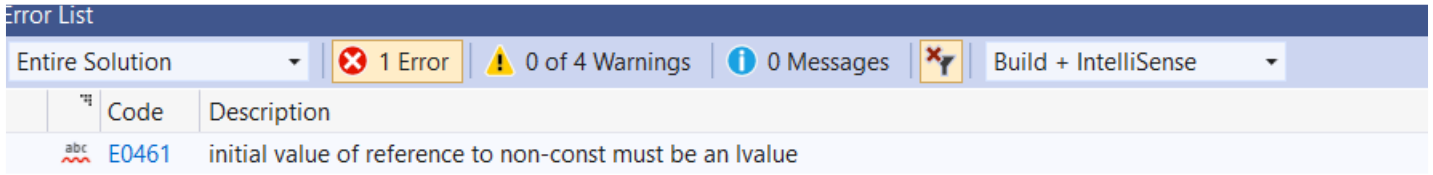
MOVE КОНСТРУКТОР

- Подаване на стойност по референция

```
void printValue(int& value) {
  //Some magical code...
}

int main() {
  printValue(int(5));

  return 0;
}
```



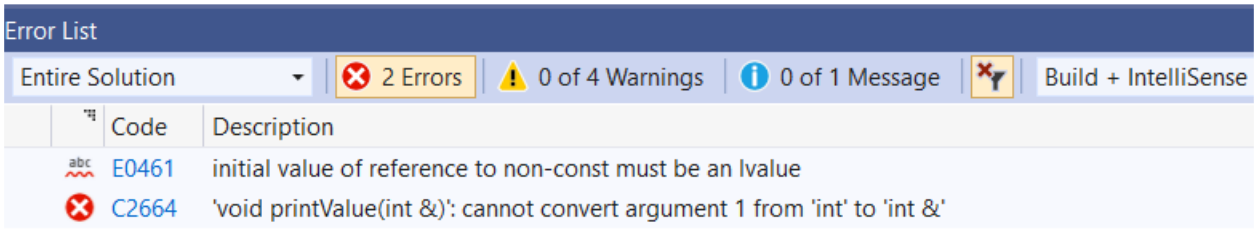
MOVE КОНСТРУКТОР

- Подаване на стойност по референция

```
void printValue(int& value) {
  //Some magical code...
}

int main() {
  int val = 5;
  printValue(val++);

  return 0;
}
```



MOVE КОНСТРУКТОР

- Подаване на стойност по референция

```
void printValue(int& value) {
  //Some magical code...
}

int main() {
  int val = 5;
  printValue(++val); // Валидно извикване

  return 0;
}
```

MOVE КОНСТРУКТОР

- Подаване на стойност по референция

```
void printValue(int& value) {
  //Some magical code...
}

int main() {
  int val = 5;
  // Валидно извикване, но е memory leak
  printValue(*(new int));

return 0;
}
```

MOVE КОНСТРУКТОР

- && - за стойности без адрес в паметта

```
void printValue(int&& value) {
  // Some magical code...
}

int main() {
  int val = 10;     // Заделяне на памет
  printValue(val); // Грешка при компилация
  printValue(5);   // Предаване на фактически параметър
  printValue(val+3); // Предаване на фактически параметър

  int&& newValName = val; // Грешка при компилация
  int&& newFiveName = 5;  // Коректно присвояване

  return 0;
}
```

MOVE КОНСТРУКТОР

- && - за стойности без адрес в паметта
  - Тази памет няма да се използва след това
  - Как може да се възползваме от това?

Move конструктор

- Как да се възползваме от rvalue
  - Да си откраднем паметта за this, тя няма да се използва пак

MOVE КОНСТРУКТОР

```
class MemoryBlock {
public:
  // Simple constructor that initializes the resource.
  MemoryBlock(size_t length);
  // Destructor.
  ~MemoryBlock();

  // Copy constructor.
  MemoryBlock(const MemoryBlock& other);

  // Copy assignment operator.
  MemoryBlock& operator=(const MemoryBlock& other);

  // Move constructor
  MemoryBlock(MemoryBlock&& other);
  // Move assignment operator
  MemoryBlock& operator=(MemoryBlock&& other);
private:
  size_t length; // The length of the resource.
  int* data; // The resource.
};
```

Ref: https://docs.microsoft.com/en-us/cpp/cpp/move-constructors-and-move-assignment-operators-cpp?view=msvc-160

MOVE КОНСТРУКТОР

```
// Copy constructor
MemoryBlock::MemoryBlock(const MemoryBlock& other) : length(other.length) {
  // Заделяне на памет и копиране на стойностите
  data = new int[other.length];
  std::copy(other.data, other.data + length, data);
}
```

Ref: https://docs.microsoft.com/en-us/cpp/cpp/move-constructors-and-move-assignment-operators-cpp?view=msvc-160

MOVE КОНСТРУКТОР

```
// Move constructor
MemoryBlock::MemoryBlock(MemoryBlock&& other) : data(nullptr),
length(0) {
    // Копиране на информацията
    data = other.data;
    length = other.length;

    // Освобождаване на източника на данни
    other.data = nullptr;
    other.length = 0;
}
```

MOVE КОНСТРУКТОР

Пример:

```
int main() {
  std::vector<MemoryBlock> v;
  v.push_back(MemoryBlock(25));

  return 0;
}
```

MOVE КОНСТРУКТОР

- Специфики
  - Ако move конструктор не е дефиниран, програмата извиква копи конструктор

Пример:

```
int main() {
  std::vector<MemoryBlock> v;
  v.push_back(MemoryBlock(25));

  return 0;
}
```

Оператор за присвояване

```
// Copy assignment operator
MemoryBlock& MemoryBlock::operator=(const MemoryBlock& other) {
  if (this != &other) {
    // Освобождаване на заделената памет
    delete [] data;

    // Копиране на информацията
    length = other.length;
    data = new int[length];
    std::copy(other.data, other.data + length, data);
  }
  return *this;
}
```

MOVE ОПЕРАТОР ЗА ПРИСВОЯВАНЕ

```
// Move assignment operator
MemoryBlock& MemoryBlock::operator=(MemoryBlock&& other) {
  if (this != &other) {
    // Освобождаване на заделената памет
    delete[] data;

    // Копиране на информацията
    data = other.data;
    length = other.length;

    // Освобождаване на източника на данни
    other.data = nullptr;
    other.length = 0;
  }
  return *this;
}
```

MOVE КОНСТРУКТОР

Пример:

```
int main() {
  MemoryBlock obj1(25);
  MemoryBlock obj2(15);
  obj1 = obj2;
  obj1 = MemoryBlock(10);

  return 0;
}
```

MOVE КОНСТРУКТОР

- Специфики
  - Ако move оператор не е дефиниран, програмата извиква копи оператора

Пример:

```
int main() {
  MemoryBlock obj1(25);
  MemoryBlock obj2(15);
  obj1 = obj2;
  obj1 = MemoryBlock(10);

  return 0;
}
```

MOVE КОНСТРУКТОР

- Специфики
  - Rvalue не може да се предава по рефенция(без запазената дума const)

```
void testParamFunc(MemoryBlock &block) {
    // Some magical code...
}

int main() {
    testParamFunc(MemoryBlock(25));
    return 0;
```

Compilation failed due to following error(s).

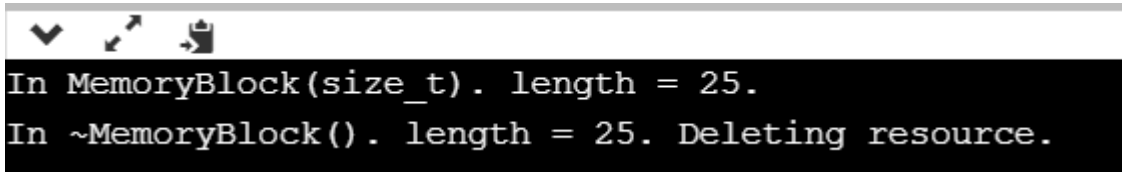
```
main.cpp: In function 'int main()':
main.cpp:126:17: error: invalid initialization of non-const reference of type 'MemoryBlock&' from an rvalue of type 'MemoryBlock'
   testParamFunc(MemoryBlock(25));
                 ^~~~~~~~~~~~~~~
```

MOVE КОНСТРУКТОР

- Специфики
  - Ако се подава rvalue по стойност, копирането на обекта се пропуска

```
void testParamFunc(MemoryBlock block) {
    // Some magical code...
}

int main() {
    testParamFunc(MemoryBlock(25));
    return 0;
}
```

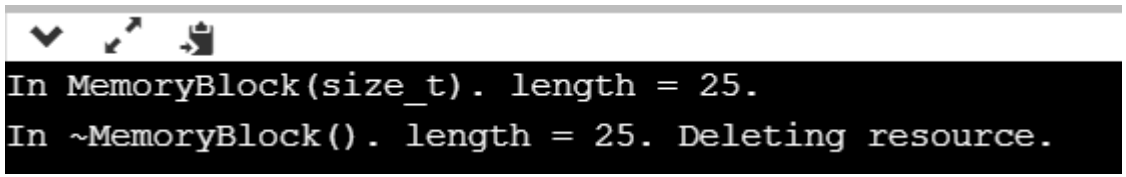


MOVE КОНСТРУКТОР

- Специфики
  - Ще работи, но копирането ще се прескочи отново

```
void testParamFunc(MemoryBlock &&block) {
    // Some magical code...
}

int main() {
    testParamFunc(MemoryBlock(25));
    return 0;
}
```



MOVE КОНСТРУКТОР

- Copy elision (пропускане на копирането - https://en.cppreference.com/w/cpp/language/copy_elision)
  - При връщане на rvalue от същия тип

```
MemoryBlock testParamFunc(MemoryBlock block) {
   return MemoryBlock(25);
}
```

  - При инициализация с rvalue от същия тип

```
MemoryBlock result = MemoryBlock(25);
```

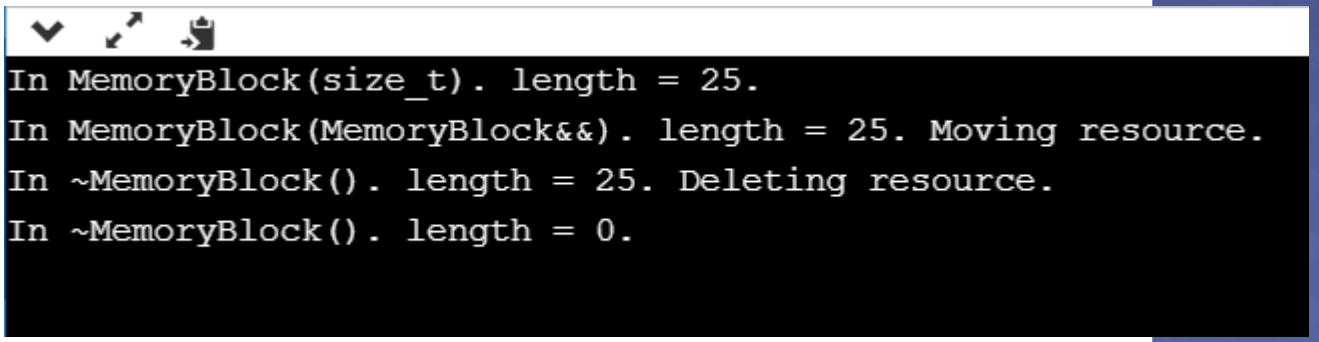
  - и други

MOVE КОНСТРУКТОР

- Явно извикване - std::move

```
void testParamFunc(MemoryBlock block) {
  // Some magical code...
}


int main() {
  MemoryBlock blockObj(25);
  testParamFunc(std::move(blockObj));
  return 0;
}
```

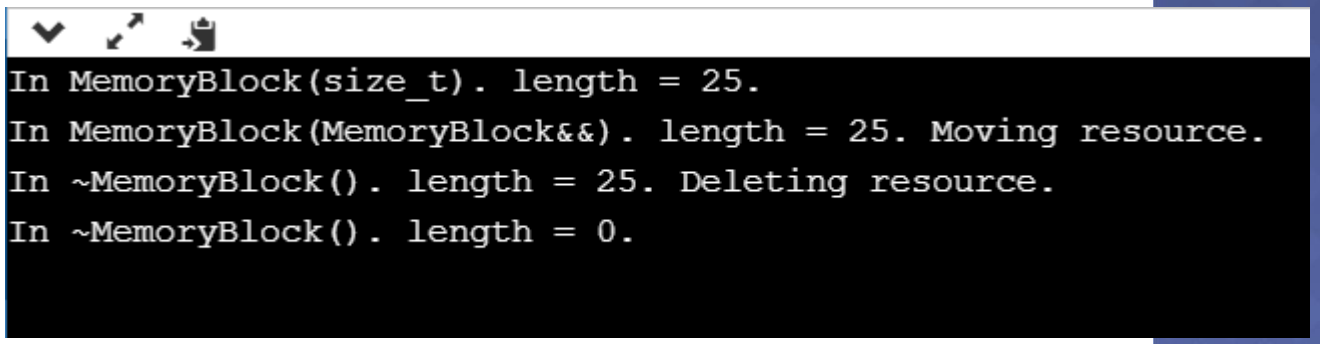


MOVE КОНСТРУКТОР

- Явно извикване - std::move

```
void testParamFunc(MemoryBlock block) {
  // Some magical code...
}


int main() {
  testParamFunc(std::move(MemoryBlock(25)));
  return 0;
}
```



ВРЕМЕ ЗА ВАШИТЕ ВЪПРОСИ