

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Programming in C++ - Primer

Lesson 7 - Typecasts, Iterators & Templates

Jakub 'Eremiell' Marek
<marekj14@fel.cvut.cz>

Silicon Hill C++ Academy

2013/12/02

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

1 Revision

2 Handing over

3 Typecasts

4 Containers

5 Iterators

6 Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Objects

C++ Primer

Jakub Marek

Revision

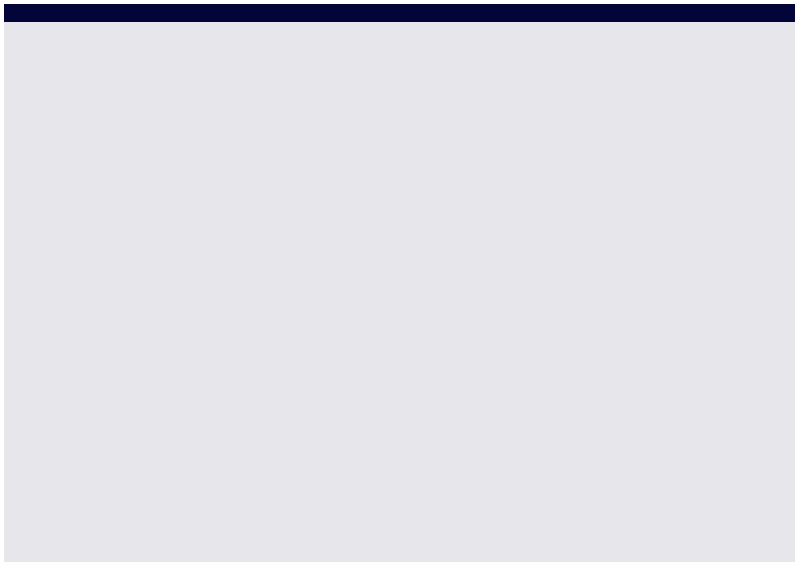
Handing over

Typecasts

Containers

Iterators

Templates



C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it
- includes whatever is needed

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file
- includes the header file

Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file
- includes the header file
- is usually the only include

Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file
- includes the header file
- is usually the only include

Access rights

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file
- includes the header file
- is usually the only include

Access rights

- public

Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file
- includes the header file
- is usually the only include

Access rights

- public
- protected

Declaration

- header file
- included wherever we use it
- includes whatever is needed

Definition

- source file
- includes the header file
- is usually the only include

Access rights

- public
- protected
- private

Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety
- getters/setters

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

Operators

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

Operators

- can be overloaded

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

Operators

- can be overloaded
- beware of member/non-member operators!

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

Operators

- can be overloaded
- beware of member/non-member operators!

Friendship

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

Operators

- can be overloaded
- beware of member/non-member operators!

Friendship

- breaks encapsulation

Encapsulation

- helps data safety
- getters/setters

Inheritance & Polymorphism

- related classes should inherit
- specialization/generalization
- early/late binding

Operators

- can be overloaded
- beware of member/non-member operators!

Friendship

- breaks encapsulation
- helps divided abstractions

Handing over Objects

C++ Primer

Jakub Marek

Revision

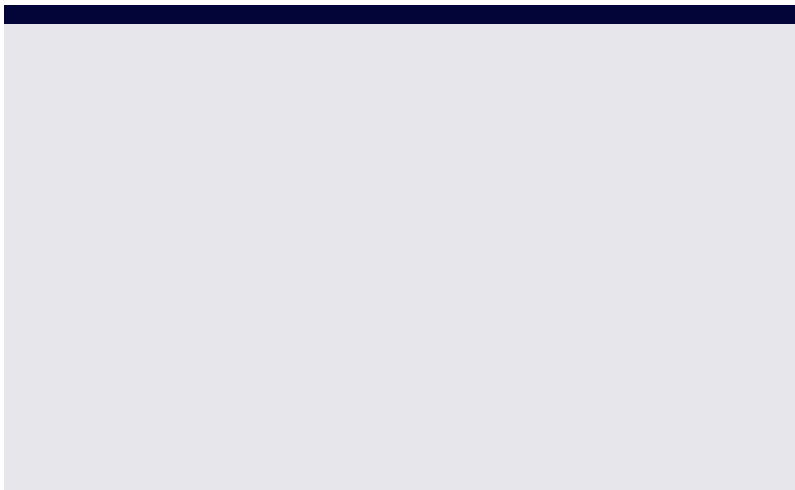
Handing over

Typecasts

Containers

Iterators

Templates



Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

to pass by value means to copy

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

to pass by value means to copy
you've got to have

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

to pass by value means to copy
you've got to have

- copy ctor

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

to pass by value means to copy
you've got to have

- copy ctor
- copy assignment operator

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

to pass by value means to copy
you've got to have

- copy ctor
- copy assignment operator
- dtor

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

once again three possibilities

- by value
- by pointer
- by reference

to pass by value means to copy
you've got to have

- copy ctor
- copy assignment operator
- dtor

The Rule of Three

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

passing by pointers & references gives original

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

passing by pointers & references gives original
you can't change reference

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

passing by pointers & references gives original
you can't change reference
you have to initialize reference at creation

Handing over Objects

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

passing by pointers & references gives original
you can't change reference
you have to initialize reference at creation
reference cannot be NULL

Questions?

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

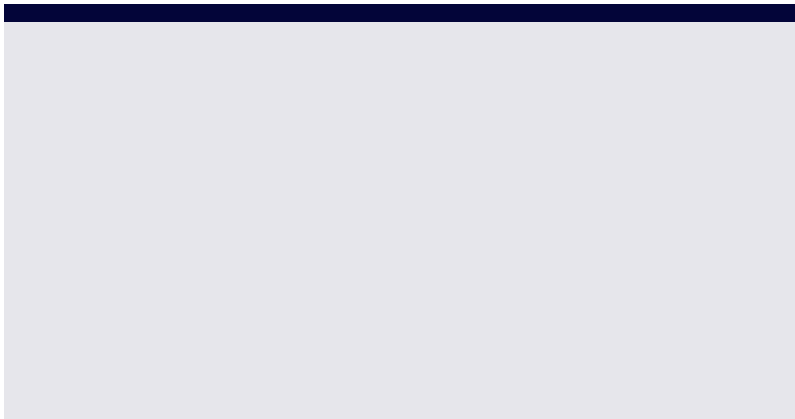
Handing over

Typecasts

Containers

Iterators

Templates



Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

we've already seen:

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

we've already seen:

- implicit typecasting

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

we've already seen:

- implicit typecasting

```
int i = 'b';
```

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

we've already seen:

- implicit typecasting

```
int i = 'b';
```

- explicit typecasting

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

we've already seen:

- implicit typecasting

```
int i = 'b';
```

- explicit typecasting

```
std::cout << (char) i << std::endl;  
std::cout << char(i) << std::endl;
```

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

we've already seen:

- implicit typecasting

```
int i = 'b';
```

- explicit typecasting

```
std::cout << (char) i << std::endl;  
std::cout << char(i) << std::endl;
```

works seamlessly only with plain old data (primitives)

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

four new typecasts

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

four new typecasts
templates

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

four new typecasts
templates
works on pointers & references

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ `dynamic_cast`

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

- `dynamic_cast`

most powerful

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

- `dynamic_cast`

most powerful

only successful if you're casting into right class

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ `dynamic_cast`

most powerful

only successful if you're casting into right class

```
Animal *a = new Cat("Moonlight");  
Cat *c = dynamic_cast<Cat*>(a);
```

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

- `dynamic_cast`

most powerful

only successful if you're casting into right class

```
Animal *a = new Cat("Moonlight");  
Cat *c = dynamic_cast<Cat*>(a);
```

- `static_cast`

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ `dynamic_cast`

most powerful

only successful if you're casting into right class

```
Animal *a = new Cat("Moonlight");  
Cat *c = dynamic_cast<Cat*>(a);
```

■ `static_cast`

only checks for compatibility

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ `dynamic_cast`

most powerful

only successful if you're casting into right class

```
Animal *a = new Cat("Moonlight");  
Cat *c = dynamic_cast<Cat*>(a);
```

■ `static_cast`

only checks for compatibility

ie. are the types related to each other?

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ `dynamic_cast`

most powerful

only successful if you're casting into right class

```
Animal *a = new Cat("Moonlight");  
Cat *c = dynamic_cast<Cat*>(a);
```

■ `static_cast`

only checks for compatibility

ie. are the types related to each other?

programmer has to take care of correctness

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ `dynamic_cast`

most powerful

only successful if you're casting into right class

```
Animal *a = new Cat("Moonlight");  
Cat *c = dynamic_cast<Cat*>(a);
```

■ `static_cast`

only checks for compatibility

ie. are the types related to each other?

programmer has to take care of correctness

```
Animal *a = new Cat("Moonlight");  
Bat *b = static_cast<Bat*>(a);
```

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ reinterpret_cast

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

- `reinterpret_cast`

will copy anything

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

- `reinterpret_cast`

will copy anything
usually platform dependent

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

- `reinterpret_cast`

will copy anything
usually platform dependent
you can store pointers as integers etc.

■ reinterpret_cast

will copy anything

usually platform dependent

you can store pointers as integers etc.

```
Animal *a = new Cat("Moonlight");  
Car *c = reinterpret_cast<Car*>(a);
```

Advanced Typecasts

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

■ reinterpret_cast

will copy anything

usually platform dependent

you can store pointers as integers etc.

```
Animal *a = new Cat("Moonlight");  
Car *c = reinterpret_cast<Car*>(a);
```

■ const_cast

■ reinterpret_cast

will copy anything

usually platform dependent

you can store pointers as integers etc.

```
Animal *a = new Cat("Moonlight");  
Car *c = reinterpret_cast<Car*>(a);
```

■ const_cast

casts const objects into non-const and vice versa

■ reinterpret_cast

will copy anything
usually platform dependent
you can store pointers as integers etc.

```
Animal *a = new Cat("Moonlight");
Car *c = reinterpret_cast<Car*>(a);
```

■ const_cast

casts const objects into non-const and vice versa

```
int encode(char c) {...}
const char c = 'x';
std::cout << encode(const_cast<char*>(c));
```

Questions?

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

vector dynamic array, adding to the end

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

vector dynamic array, adding to the end
deque dynamic array, adding to both sides

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

vector dynamic array, adding to the end
deque dynamic array, adding to both sides
list linked list, a lot of adding

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

`vector` dynamic array, adding to the end
`deque` dynamic array, adding to both sides
`list` linked list, a lot of adding
`set` set (only unique members), no order
`multiset` set with duplicities

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

`vector` dynamic array, adding to the end
`deque` dynamic array, adding to both sides
`list` linked list, a lot of adding
`set` set (only unique members), no order
`multiset` set with duplicities
`map` mapping given key to a value, unique keys
`multimap` duplicate keys allowed

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

`vector` dynamic array, adding to the end

`deque` dynamic array, adding to both sides

`list` linked list, a lot of adding

`set` set (only unique members), no order

`multiset` set with duplicities

`map` mapping given key to a value, unique keys

`multimap` duplicate keys allowed

`stack` stack LIFO

`queue` queue FIFO

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

`vector` dynamic array, adding to the end

`deque` dynamic array, adding to both sides

`list` linked list, a lot of adding

`set` set (only unique members), no order

`multiset` set with duplicities

`map` mapping given key to a value, unique keys

`multimap` duplicate keys allowed

`stack` stack LIFO

`queue` queue FIFO

`priority_queue` queue with priority

Containers Revisited

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

`vector` dynamic array, adding to the end

`deque` dynamic array, adding to both sides

`list` linked list, a lot of adding

`set` set (only unique members), no order

`multiset` set with duplicities

`map` mapping given key to a value, unique keys

`multimap` duplicate keys allowed

`stack` stack LIFO

`queue` queue FIFO

`priority_queue` queue with priority

`string` a string

`rope` a string container

Questions?

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Iterators

C++ Primer

Jakub Marek

Revision

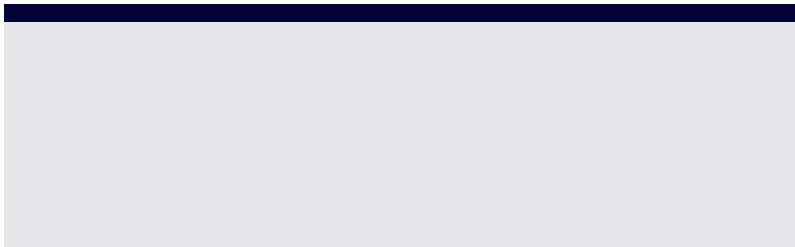
Handing over

Typecasts

Containers

Iterators

Templates



Iterators

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allows you to traverse containers

Iterators

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allows you to traverse containers
several types of

Iterators

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allows you to traverse containers
several types of
each container has it's own

allows you to traverse containers
several types of
each container has it's own

```
std::vector<Animal> a;  
std::vector<Animal>::iterator it;  
for (it = a.begin(); it != a.end() && newpet > *it; it++) {}  
a.insert(it, newpet);
```

Questions?

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Templates

C++ Primer

Jakub Marek

Revision

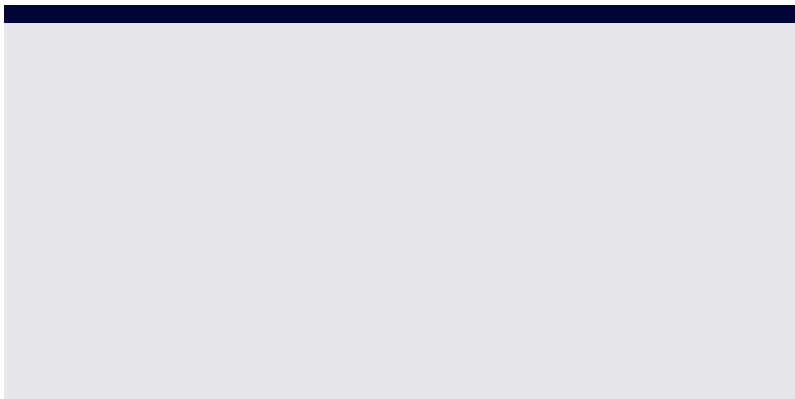
Handing over

Typecasts

Containers

Iterators

Templates



Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allow you to make classes or functions

Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allow you to make classes or functions
which work dynamically with different other classes

Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allow you to make classes or functions
which work dynamically with different other classes
or you can pass them any parameters

Templates

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

allow you to make classes or functions
which work dynamically with different other classes
or you can pass them any parameters

```
template<class T>  
template<class T = int>  
template<int N>  
template<class T = char, int N = 10>  
int func<T>(T param) {...}  
class c {T a, b, c; T[N] ar;};
```

Template Specialization

C++ Primer

Jakub Marek

Revision

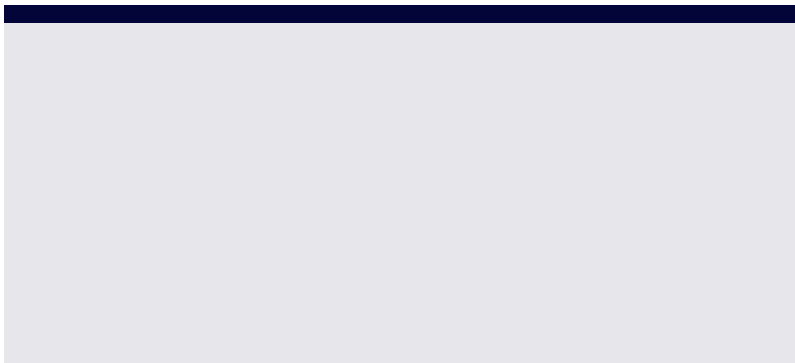
Handing over

Typecasts

Containers

Iterators

Templates



Template Specialization

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

oftentimes template does just fine for most classes

Template Specialization

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

oftentimes template does just fine for most classes
but not for some...

Template Specialization

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

oftentimes template does just fine for most classes
but not for some...
then, we can specialize

Template Specialization

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

oftentimes template does just fine for most classes
but not for some...

then, we can specialize

```
template<class T>  
class myobject {T ar[80];};
```

```
template<>  
class myobject<bool> {unsigned char ar[10];};
```


Questions?

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates

Break!

C++ Primer

Jakub Marek

Revision

Handing over

Typecasts

Containers

Iterators

Templates