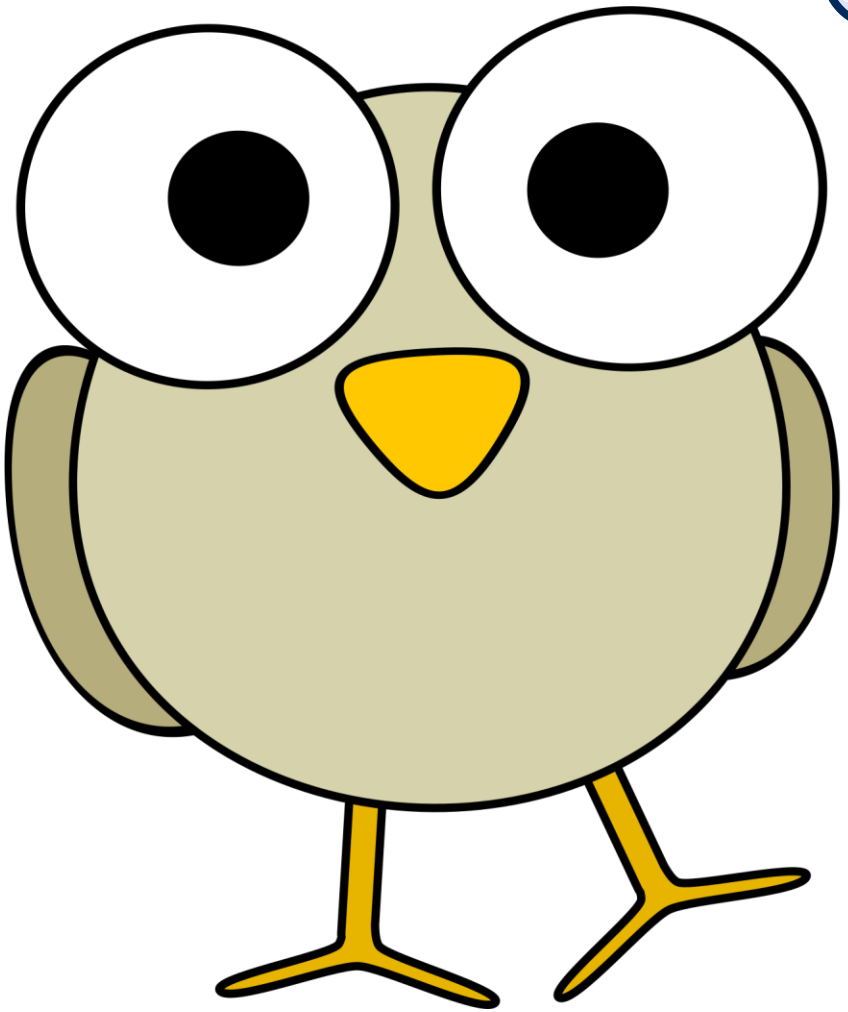
CSCI 200: Foundational Programming Concepts & Design Lecture 23

Templated Classes

Previously in CSCI 200

- The Big 3
  - Copy Constructor
  - Copy Assignment Operator
  - Destructor
- Shallow Copy & Deep Copy

Questions?



OOP Quiz

- Make Canvas Full Screen
- Access Code:
- 12 Minutes

Learning Outcomes For Today

- Discuss the benefits of templated classes and functions

On Tap For Today

- Templated Functions
- Templated Classes
- Practice

On Tap For Today

- Templated Functions
- Templated Classes
- Practice

To Do: `int maxt(int, int)`

- Write a function to return the larger of two integers

To Do: `float maxt(float, float)`

- Write a function to return the larger of two floats

To Do: `double maxt(double, double)`

- Write a function to return the larger of two doubles

To Do: `char maxt(char, char)`

- Write a function to return the larger of two characters

Comparing Functions

- What's the same / different between each implementation?

```
int    maxt(const int    LHS, const int    RHS) { return LHS < RHS ? RHS : LHS; }
float  maxt(const float  LHS, const float  RHS) { return LHS < RHS ? RHS : LHS; }
double maxt(const double LHS, const double RHS) { return LHS < RHS ? RHS : LHS; }
char   maxt(const char   LHS, const char   RHS) { return LHS < RHS ? RHS : LHS; }
```

Create a Template!

- Abstract out the data type
- Have seen already with _____?

Create a Template!

- Abstract out the data type
- Have seen already with vector.

```
vector<int> intVector;
intVector.push_back( 1 );

vector<float> floatVector;
floatVector.push_back( 1.0f );

vector<string> stringVector;
stringVector.push_back( "1" );

vector<Zombie*> zombieVector;
zombieVector.push_back( new Zombie );
```

Templated Functions

- Use a variable datatype

```
template<typename T>
T maxt(const T LHS, const T RHS) { return LHS < RHS ? RHS : LHS; }
```

Proper Abstraction

```
// max.h
template<typename T>
T maxt(const T LHS, const T RHS);
```

```
// max.cpp
#include "max.h"
template<typename T>
T maxt(const T LHS, const T RHS) {
    return LHS < RHS ? RHS : LHS;
}
```

```
// main.cpp
#include "max.h"

  int largerInt = maxt(4, 7);
  float largerFloat = maxt(14.3f, 5.2f);
```

However...

- If we did

```
1. g++ -o max.o -c max.cpp
2. g++ -o main.o -c main.cpp
3. g++ -o Lec23.exe max.o main.o
```

- When is template type `T` known?

From This

```
// max.h
template<typename T>
T maxt(const T LHS, const T RHS);
```

```
// max.cpp
#include "max.h"
template<typename T>
T maxt(const T LHS, const T RHS) {
    return LHS < RHS ? RHS : LHS;
}
```

```
// main.cpp
#include "max.h"

  int largerInt = maxt(4, 7);
  float largerFloat = maxt(14.3f, 5.2f);
```

To This

```
// max.hpp
template<typename T>
T maxt(const T LHS, const T RHS);


//--function implementations--
template<typename T>
T maxt(const T LHS, const T RHS) {
    return LHS < RHS ? RHS : LHS;
}
```

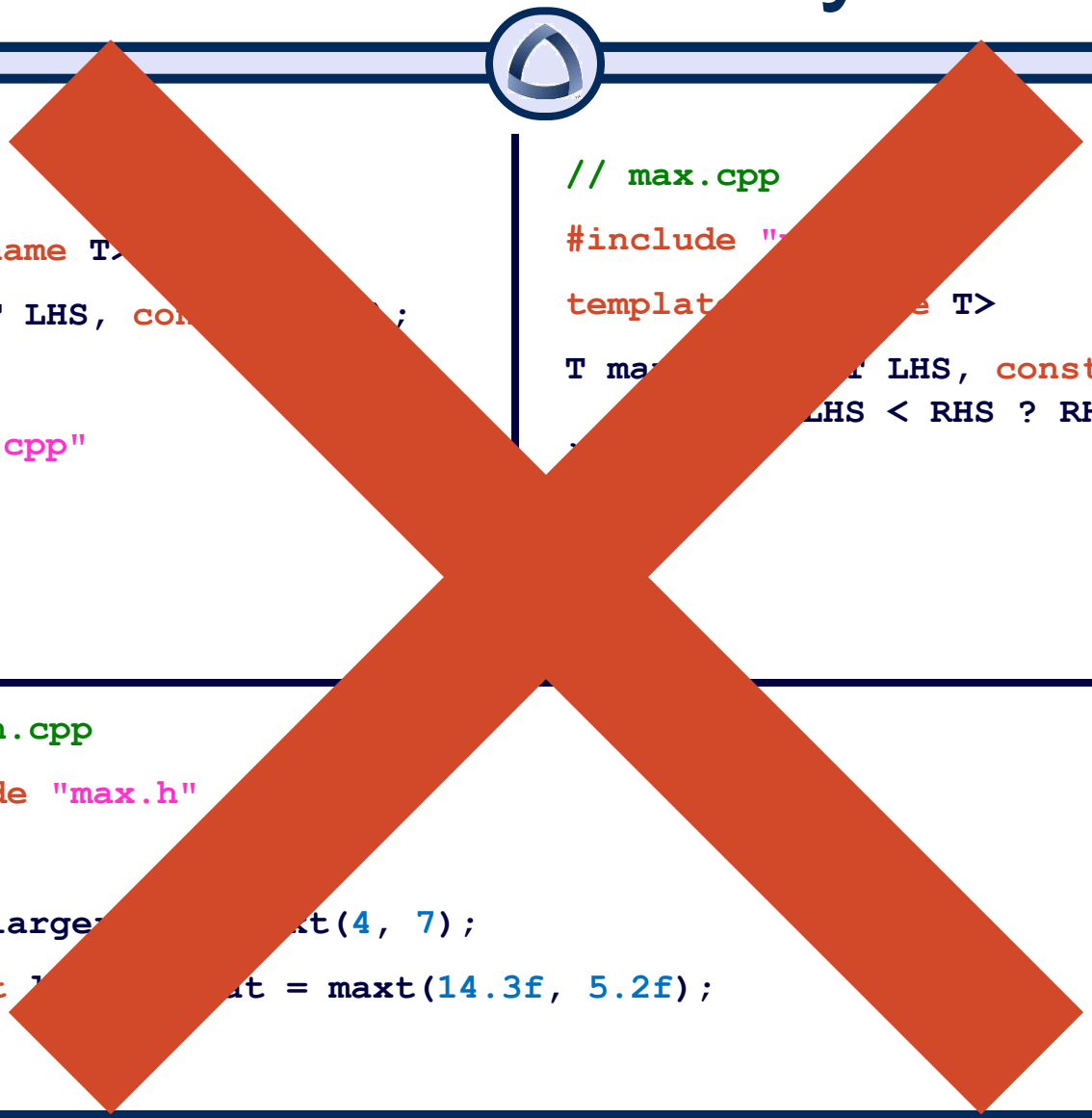
```
// main.cpp
#include "max.hpp"


  int largerInt = maxt(4, 7);
  float largerFloat = maxt(14.3f, 5.2f);
```

File Extensions

- *.h – header file
  - Contains only declarations
  - Is not compiled on its own
- *.c – compilable file
  - Compiled on its own and linked into executable
- *.*pp – implementation file
  - Contains definitions
- *.hpp – header implementation file
  - Contains declarations and corresponding definitions
  - Is not compiled on its own, still a header file
- *.cpp – compilable implementation file
  - Contains function definitions
  - Is compiled

Some Tutorials Say To Do



```
// max.h
template<typename T>
T maxt(const T LHS, co                ;

#include "max.cpp"
```

```
// max.cpp
#include "
templat                e T>
T ma                 f LHS, const T RHS) {
                LHS < RHS ? RHS : LHS;
```

```
// main.cpp
#include "max.h"

  int large          xt(4, 7);
  float              t = maxt(14.3f, 5.2f);
```

Instead Use *.hpp

```
// max.hpp
template<typename T>
T maxt(const T LHS, const T RHS);


//--function implementations--
template<typename T>
T maxt(const T LHS, const T RHS) {
    return LHS < RHS ? RHS : LHS;
}
```

```
// main.cpp
#include "max.hpp"


  int largerInt = maxt(4, 7);
  float largerFloat = maxt(14.3f, 5.2f);
```

Build Now Looks Like

```
1. g++ -o main.o -c main.cpp
2. g++ -o Lec23.exe main.o
```

On Tap For Today

- Templated Functions
- Templated Classes
- Practice

Example Box Class

```
// Box.h
class Box {
public:
    Box(const int SIZE);
    int getBoxSize() const;
private:
    int _size;
};
```

```
// Box.cpp
#include "Box.h"

Box::Box(const int SIZE) {
  _size = SIZE;
}
int Box::getBoxSize() const {
  return _size;
}
```

Add A Single Int As Content

```
// Box.h
class Box {
public:
    Box(const int SIZE);
    int getBoxSize() const;
    void putIn(const int);
    int takeOut();
private:
    int _size;
    int* _pContent;
};
```

```
// Box.cpp
#include "Box.h"

Box::Box(const int SIZE) {
  _size = SIZE;
  _pContent = nullptr;
}

int Box::getBoxSize() const {
  return _size;
}

void Box::putIn(const int VAL) {
  if(_pContent != nullptr)
    _pContent = new int(VAL);
}

int Box::takeOut() {
  if( _pContent != nullptr ) {
    int val = *_pContent;
    delete _pContent; _pContent = nullptr;
    return val;
  }
  return 0;
}
```

IntBox

- How to make a Box that holds a:
  - `int`?
  - `float`?
  - `string`?
  - `Zombie`?
- What's the same? Different?

Templated Class

```
// Box.hpp
template<typename T>
class LootBox {
public:
    LootBox(const int SIZE);
    int getBoxSize() const;
    void putIn(const T);
    T takeOut();
private:
    int _size;
    T* _pContent;
};


//--continues on next column--
```

```
//---function implementations---
template<typename T>
LootBox<T>::LootBox(const int SIZE) {
  _size = SIZE;
  _pContent = nullptr;
}
template<typename T>
int LootBox<T>::getBoxSize() const {
  return _size;
}
template<typename T>
void LootBox<T>::putIn(const T VAL) {
  if(_pContent != nullptr)
    _pContent = new T(VAL);
}
template<typename T>
T LootBox<T>::takeOut() {
  if( _pContent != nullptr ) {
    T val = *_pContent;
    delete _pContent; _pContent = nullptr;
    return val;
  }
  return T();
}
```

Use Like Vector

```
// main.cpp

LootBox<int> *pIntBox = new LootBox<int>;
pIntBox->putIn( 5 );

LootBox<string> *pStringBox = new LootBox<string>;
pStringBox->set( "hooray" );
```

On Tap For Today

- Templated Functions
- Templated Classes
- Practice

To Do For Next Time

- Keep going on Set4
- Work on Exam II Review questions