CodeCarbonCopy: Automatic Code Reuse

Oliver Goch

What is CodeCarbonCopy and Automatic Code Reuse?

Automatic Code Reuse

- System of transplanting code from one program (donor) to another (recipient)

Automatic Code Reuse

- System of transplanting code from one program (donor) to another (recipient)
- Changes the variables to match recipient

Automatic Code Reuse

- System of transplanting code from one program (donor) to another (recipient)
- Changes the variables to match recipient
- Programs can handle many file types

Automatic Code Reuse

- System of transplanting code from one program (donor) to another (recipient)
- Changes the variables to match recipient
- Programs can handle many file types
- Internally are very different

Automatic Code Reuse

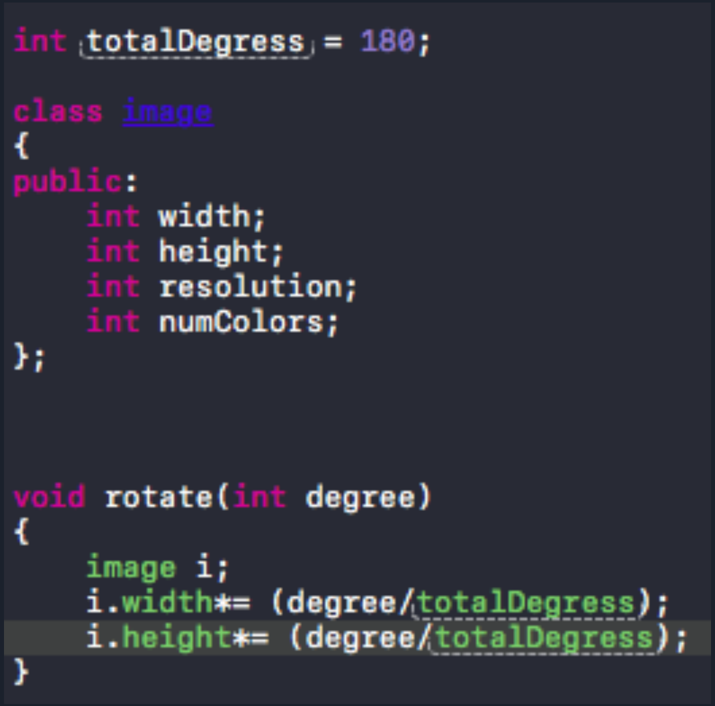
- System of transplanting code from one program (donor) to another (recipient)
- Changes the variables to match recipient
- Programs can handle many file types
- Internally are very different
- Cannot simply copy and paste functions

CodeCarbonCopy

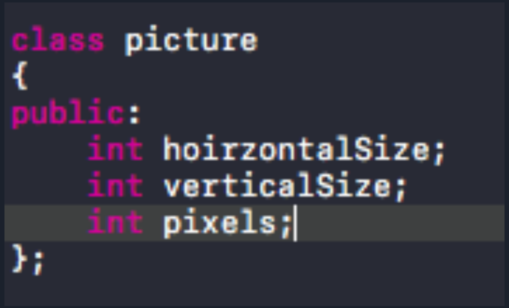
CodeCarbonCopy (CCC) is a system created by researchers at MIT’s Computer Science Artificial Intelligence Laboratory (CSAIL) that can transplant code between programs. It can recognize variables and irrelevant code in one program, and match it in another. And it does all of this *automatically*

Why is this useful?

Why?

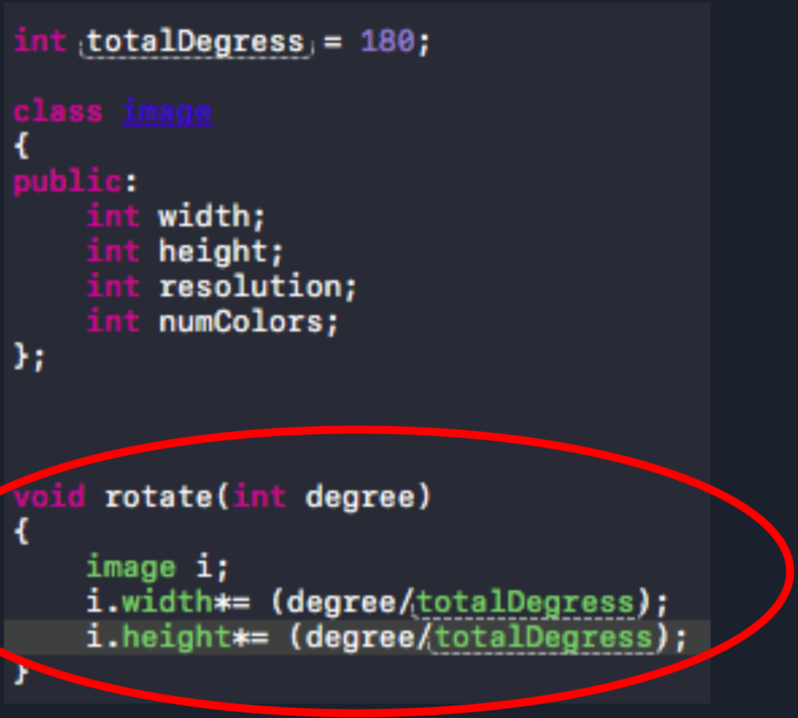


```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

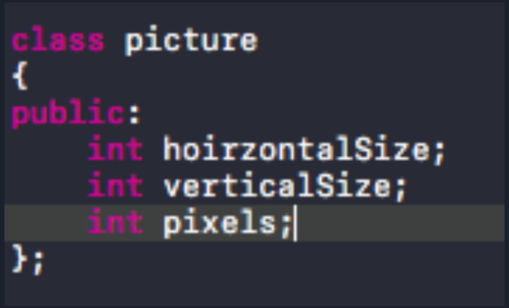


```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Why?

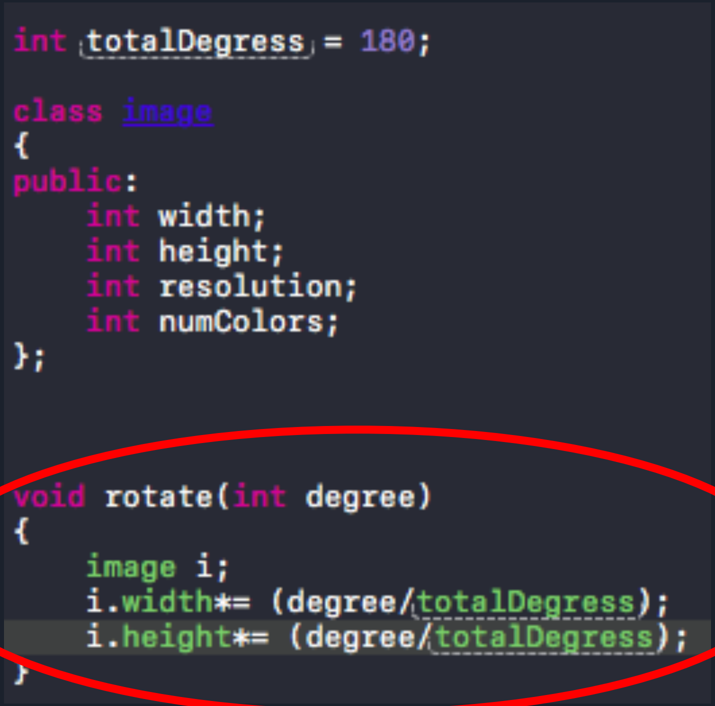


```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

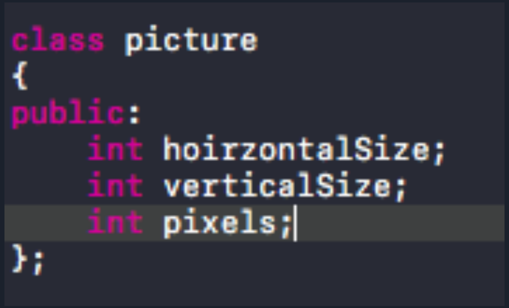


```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Why?

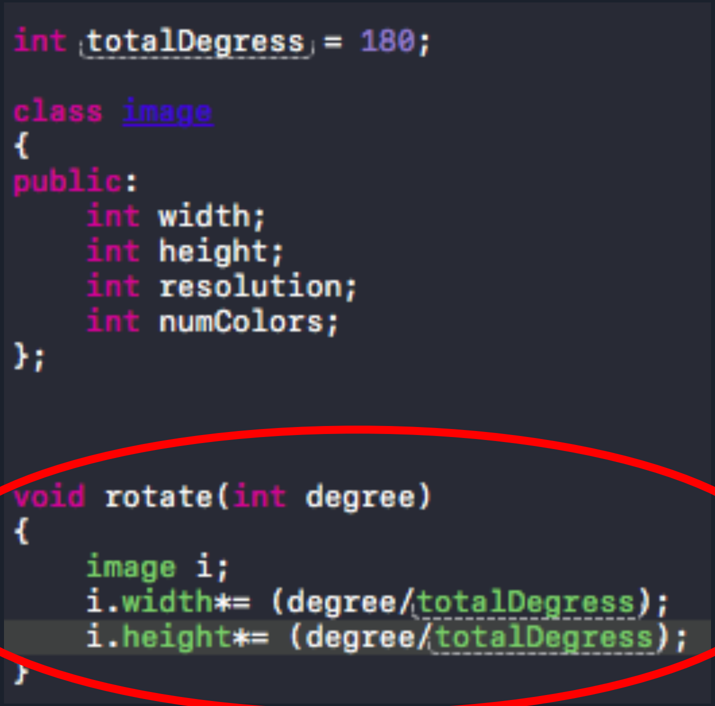


```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

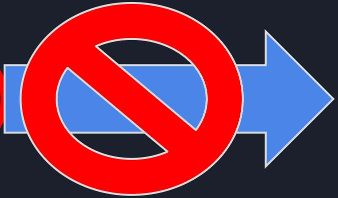


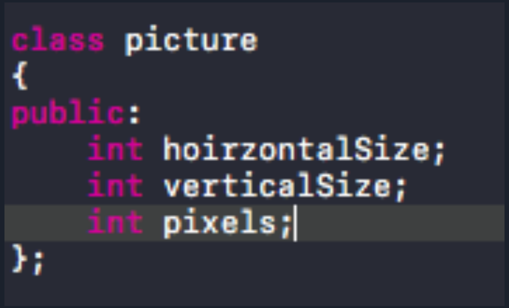
```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Why?



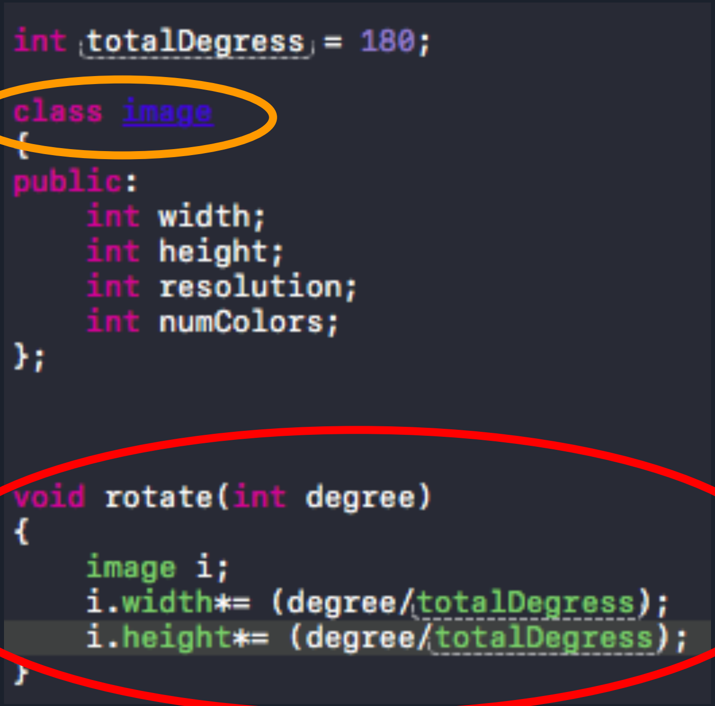
```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```



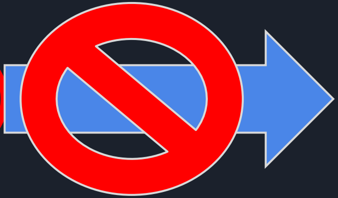


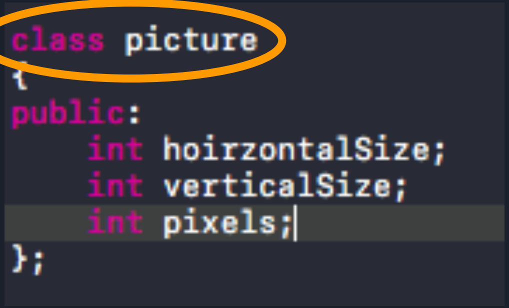
```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Why?



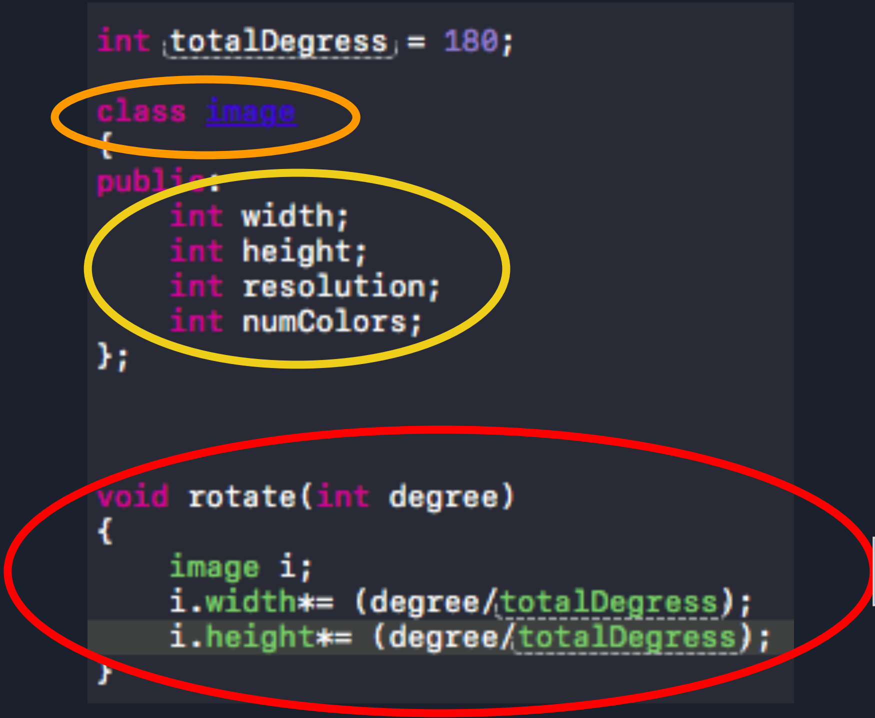
```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```



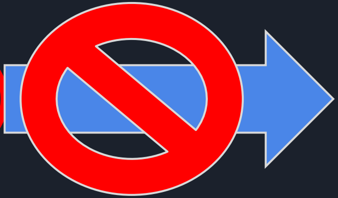


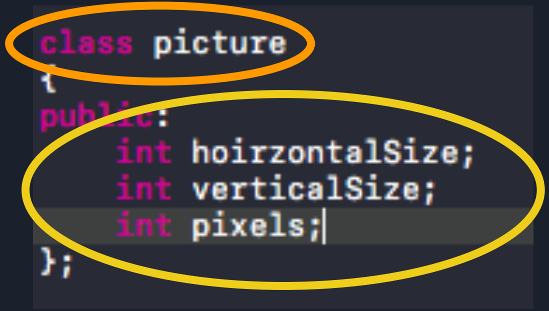
```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Why?



```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```





```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Why?

- Simple program, can just find and replace the variables

Why?

- Simple program, can just find and replace the variables
- But what about class with more variables?

Why?

- Simple program, can just find and replace the variables
- But what about class with more variables?
- But what about program with more lines?

Why?

- Simple program, can just find and replace the variables
- But what about class with more variables?
- But what about program with more lines?
- It would be impossible

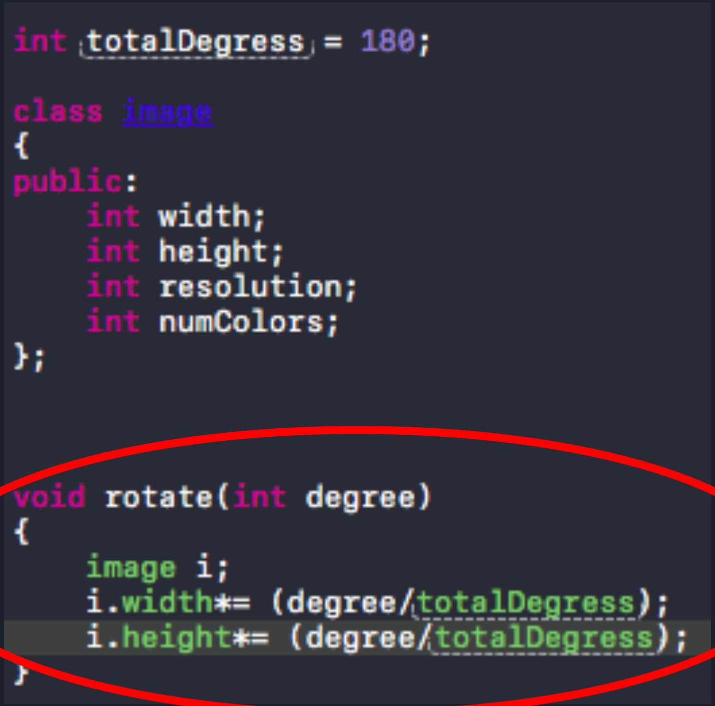
Why?

- Simple program, can just find and replace the variables
- But what about class with more variables?
- But what about program with more lines?
- It would be impossible
- This is where CCC comes in use

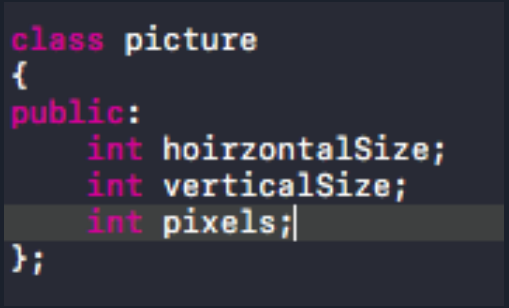
Why?

- Simple program, can just find and replace the variables
- But what about class with more variables?
- But what about program with more lines?
- It would be impossible
- This is where CCC comes in use
- CCC can do all of this automatically

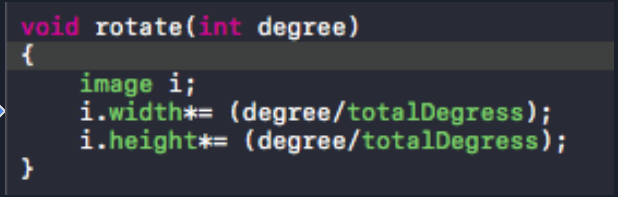
CCC in Action



```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

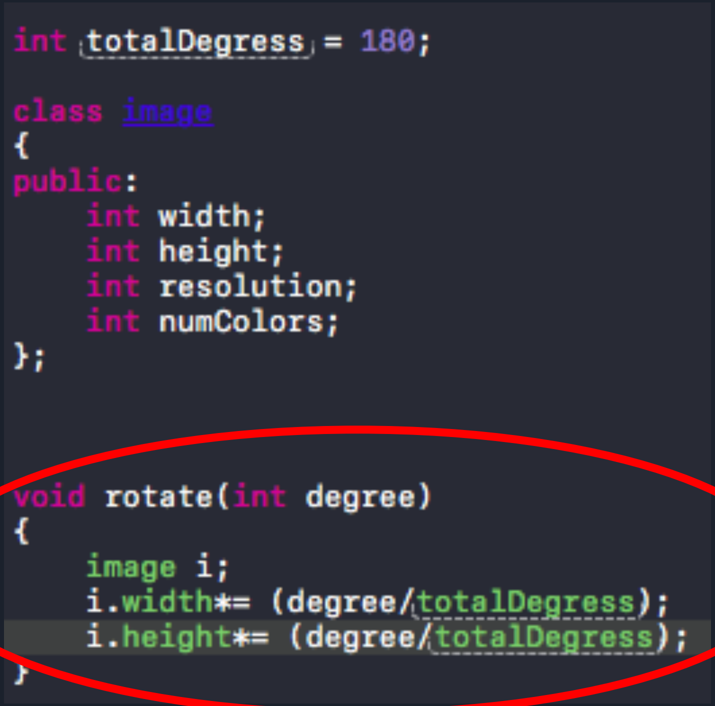


```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

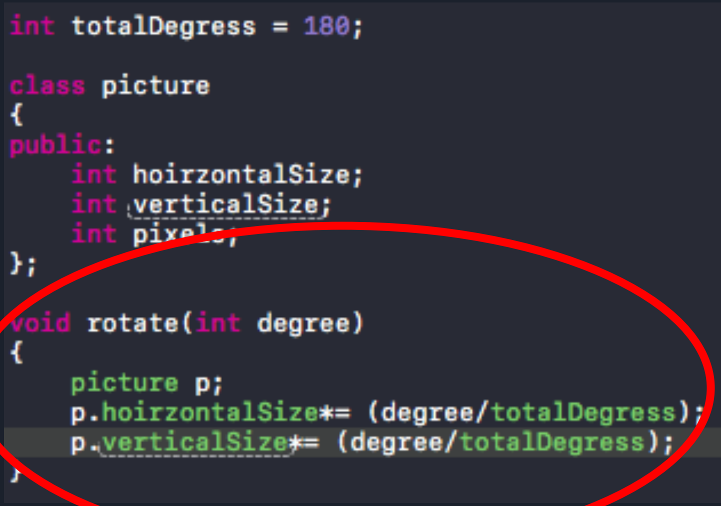


```
void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

CCC in Action

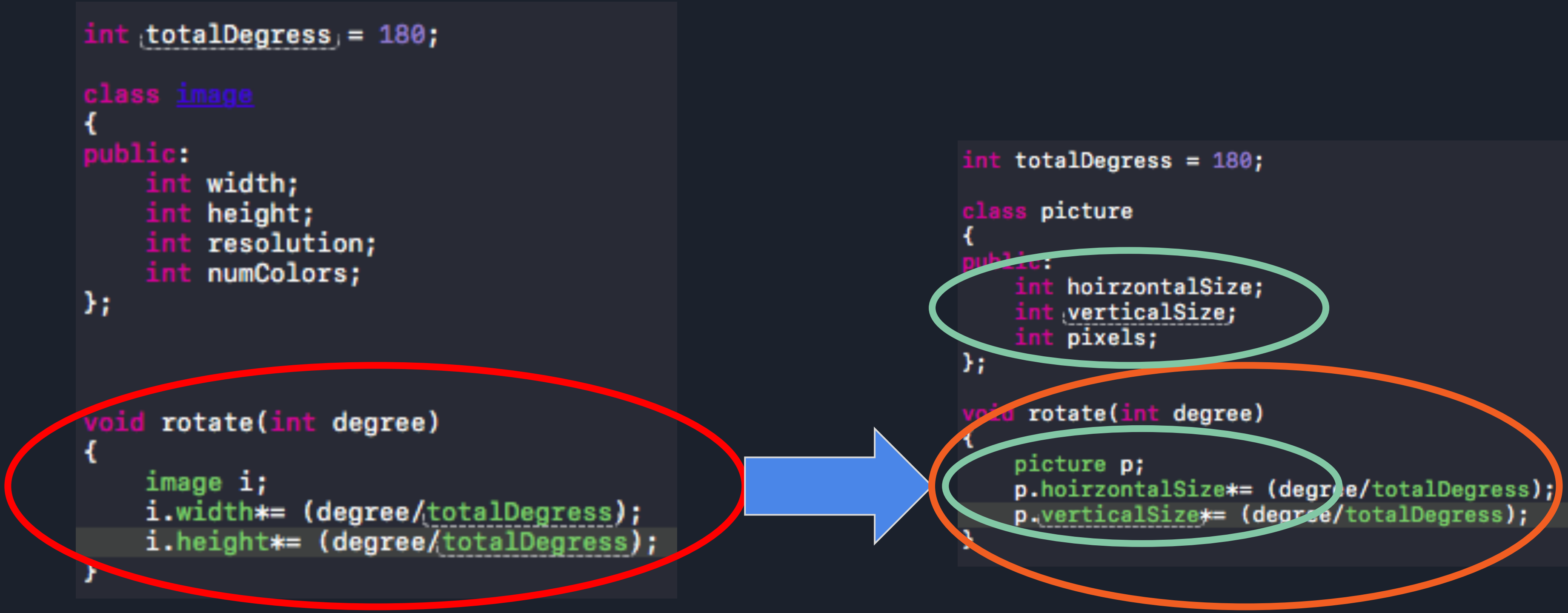


```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```



```
int totalDegress = 180;

class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};

void rotate(int degree)
{
    picture p;
    p.hoirzontalSize*= (degree/totalDegress);
    p.verticalSize*= (degree/totalDegress);
}
```

CCC in Action



```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

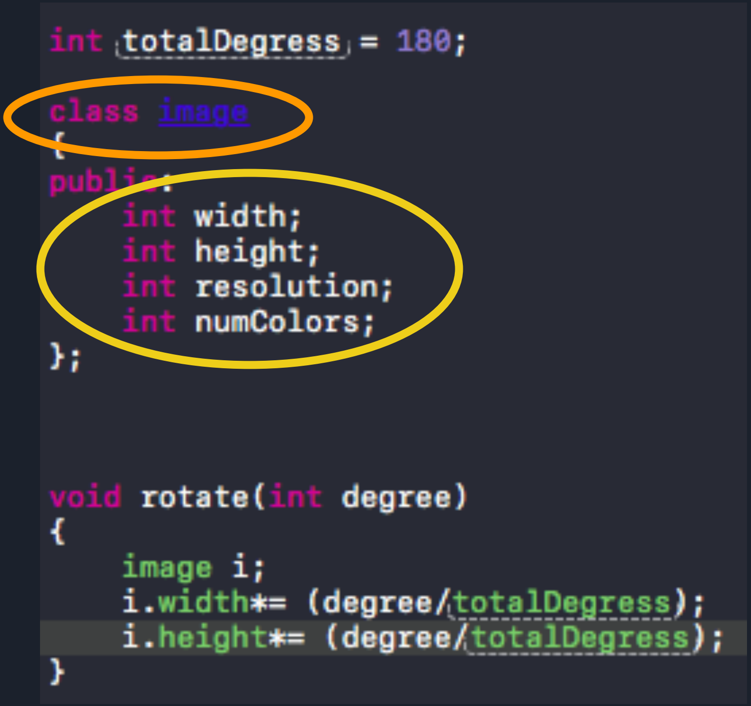
```
int totalDegress = 180;

class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};

void rotate(int degree)
{
    picture p;
    p.hoirzontalSize*= (degree/totalDegress);
    p.verticalSize*= (degree/totalDegress);
}
```

How is it done?

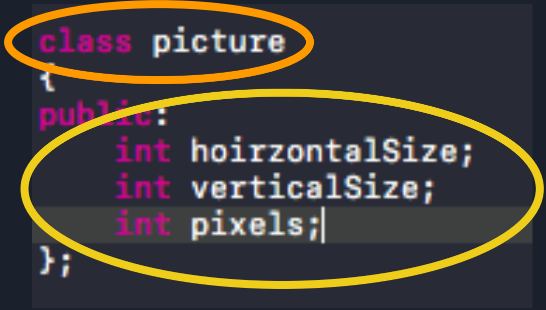
Compare program with same input file

$$\begin{tikzcd}
 & \text{image.pdf} \arrow[dl] \arrow[dr] & \\
\text{program1 (donor)} \arrow[d] & & \text{program2 (recipient)} \arrow[d] \\
\text{output1} & & \text{output2}
\end{tikzcd}$$

Finds matches and presents them to user

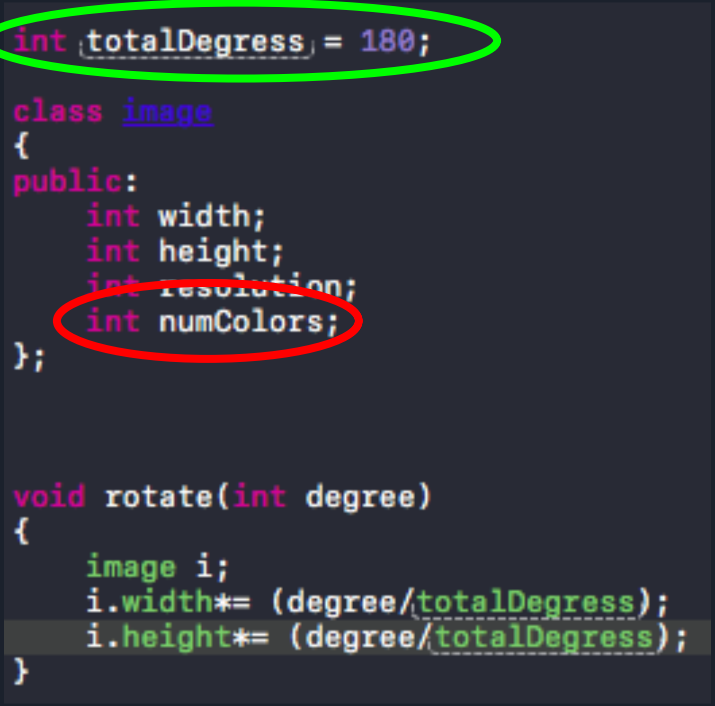


```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```

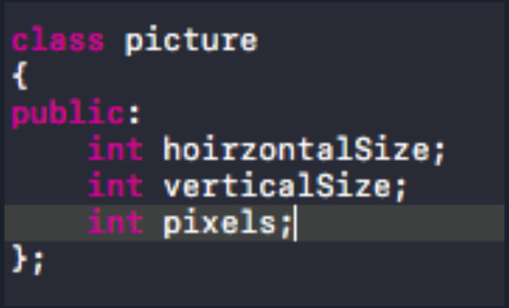


```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Shows variables not used in recipient and copies globals

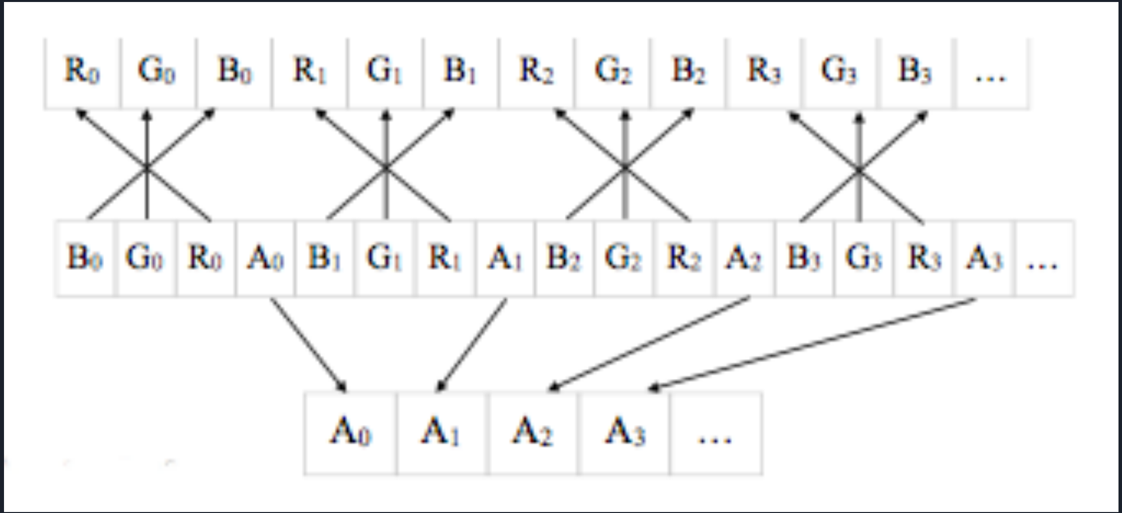


```
int totalDegress = 180;

class image
{
public:
    int width;
    int height;
    int resolution;
    int numColors;
};



void rotate(int degree)
{
    image i;
    i.width*= (degree/totalDegress);
    i.height*= (degree/totalDegress);
}
```



```
class picture
{
public:
    int hoirzontalSize;
    int verticalSize;
    int pixels;
};
```

Uses debugging to look at the specific values stored in memory in both and tries to create a relation between them


R_0 G_0 B_0 R_1 G_1 B_1 R_2 G_2 B_2 R_3 G_3 B_3 …
B_0 G_0 R_0 A_0 B_1 G_1 R_1 A_1 B_2 G_2 R_2 A_2 B_3 G_3 R_3 A_3 …
A_0 A_1 A_2 A_3 …

Conclusions

- “CodeCarbonCopy enables one of the holy grails of software engineering: automatic code reuse” - Stelios Sidiroglou-Douskos
- The technology still has a long way to go.
- It was recently unveiled in September.
- 8 tests were run on 6 open source image processing programs
- 7/8 were successful

Works Cited

Larry Hardesty | MIT News Office. “Automatic code reuse.” *MIT News*, 19 Sept. 2017, news.mit.edu/2017/automatic-code-reuse-0920.

Sidiroglou-Douskos, Stelios, et al. “CodeCarbonCopy.” *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017*, 2017, doi:10.1145/3106237.3106269.