Type Theory
in 15min

Niki Vazou
University of Maryland

The 3 Steps of Programming

Step 1: Coding

```
printf(“Hello!”); //  hello.c
```

Step 2: Compilation

```
> gcc -o hello.exec hello.c
```

Step 3: Execution

```
> ./hello.exec
> Hello!
```

The 3 Steps of Programming

Step 1: Coding

```
printf(“Hello!”); //  hello.c
```

Step 2: Compilation

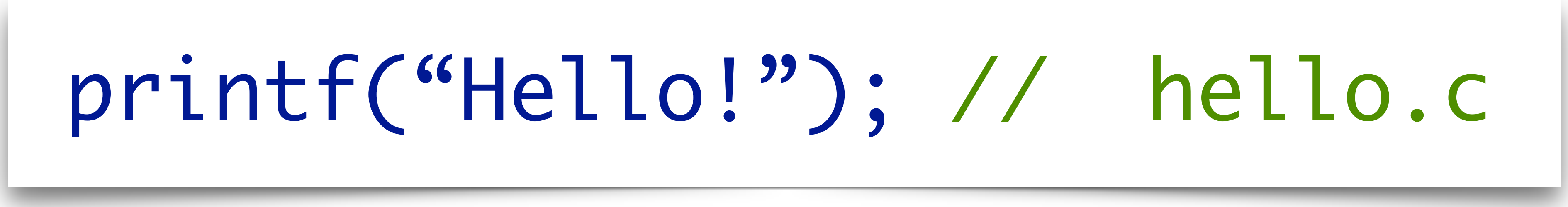
```
> gcc -o hello.exec hello.c
```

Step 3: Execution

Many times & takes long.

The 3 Steps of Programming

Step 1: Coding



```
printf(“Hello!”); //  hello.c
```

Step 2: Compilation

Few times & fast.

Step 3: Execution

Many times & takes long.

The 3 Steps of Programming

Step 1: Coding

Full of errors!

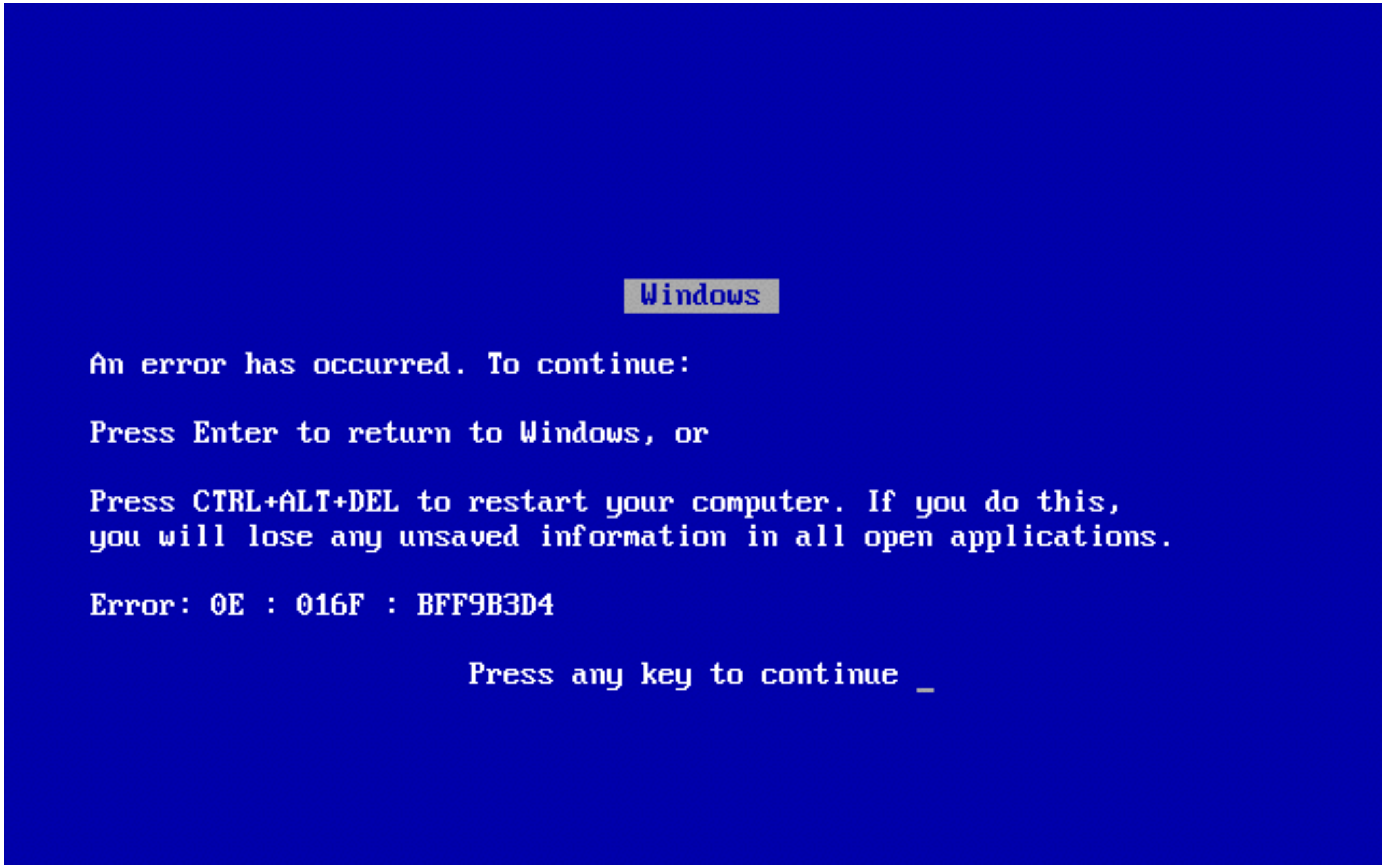
Step 2: Compilation

Few times & fast.

Step 3: Execution

Many times & takes long.

Code Errors crash at execution!



```
                              Windows

An error has occurred. To continue:

Press Enter to return to Windows, or

Press CTRL+ALT+DEL to restart your computer. If you do this,
you will lose any unsaved information in all open applications.

Error: 0E : 016F : BFF9B3D4

                       Press any key to continue _
```

Types to detect code errors!

Code Errors

```
printf(1+"Hey!");
```

What will the above print?

A.

```
> ey!
```

B.

```
> Hey!
```

C.

```
> 1+Hey!
```

D.

```
> 1Hey!
```

Code Errors

```
printf(1+"Hey!");
```

What will the above print?

A.

```
> ey!
```

Code Errors vs. Correct Code

```
printf("%d",(1+1)+2);
```

What will the above print?

A.

```
> 4
```

B.

```
> No other alternative!
```

Types

Detect Errors Early (compile time)!

Types: Classify Data

e.g., “1 has type Int”

```
1:Int
```

Types: Classify Data

```
1:Int
2:Int
```

…

```
"Hey":String
```

Types: Combine Data

If e_1 : Int and e_2 : Int, then e_1+e_2 : Int.
where e_1 and e_2 are variables.

Types: Combine Data

If e_1 : Int and e_2 : Int, then e_1+e_2 : Int.
where e_1 and e_2 are variables.

$$\frac{e_1 : \mathtt{Int} \qquad e_2 : \mathtt{Int}}{e_1 + e_2 : \mathtt{Int}}$$

Types: Example I

$$\frac{\mathtt{1}:\mathtt{Int} \qquad \mathtt{1}:\mathtt{Int}}{\mathtt{1+1}:\mathtt{Int}}$$

$e_1 := 1$

$e_2 := 1$

Types: Example II

$$\frac{\texttt{1+1:Int} \qquad \texttt{2:Int}}{\texttt{(1+1)+2:Int}}$$

e_1 := 1+1

e_2 := 2

Types: Example II

$$\dfrac{\dfrac{1:\mathtt{Int} \qquad 1:\mathtt{Int}}{1+1:\mathtt{Int}} \qquad 2:\mathtt{Int}}{(1+1)+2:\mathtt{Int}}$$

e_1 := 1+1

e_2 := 2

Types: Example III

$$\xcancel{\dfrac{1 : \mathtt{Int} \qquad \text{"Hey!"} : \mathtt{String}}{1 + \text{"Hey!"} : \,??}}$$

e_1 := 1

e_2 := "Hey!"

Types: Example III

$$\xcancel{\frac{1:\mathtt{Int} \qquad \mathtt{"Hey!"}:\mathtt{String}}{1+\mathtt{"Hey!"}:\mathtt{??}}}$$

What rules are wrong?

Soundness

If e has a type, then e cannot crash!

Definitions

Expressions e**:** written by the programmer.

e.g.,(1+1)+2

Values v**:** result of expressions.

e.g., 4

Definitions

Expressions e**:** written by the programmer.

Values v**:** result of expressions.

Evaluation $\hookrightarrow$ **:** from expressions to values.

$$(1+1)+2 \hookrightarrow 2+2 \hookrightarrow 4$$

Goes to $\hookrightarrow^{\star}$**:** many steps of evaluation.

$$(1+1)+2 \hookrightarrow^{\star} 4$$

Definitions

Expressions e**:** written by the programmer.

Values v**:** result of expressions.

Evaluation $\hookrightarrow$ **:** from expressions to values.

Goes to $\hookrightarrow^{\star}$**:** many steps of evaluation.

Crash `crash`**:** untyped crashing expression.

Soundness

Expressions e**:** written by the programmer.

Values v**:** result of expressions.

Evaluation $\hookrightarrow$ **:** from expressions to values.

Goes to $\hookrightarrow^{\star}$**:** many steps of evaluation.

Crash `crash`**:** untyped crashing expression.

If e **has a type, then** e **cannot crash!**

Soundness

Expressions e: written by the programmer.

Values v: result of expressions.

Evaluation $\hookrightarrow$: from expressions to values.

Goes to $\hookrightarrow^*$: many steps of evaluation.

Crash crash: untyped crashing expression.

If e **has a type, then** e **cannot crash!**

If e : t**, then** e $\not\hookrightarrow^*$ crash**.**

The two Steps of Soundness

Progress:

If $e_1 : t$, then $e_1 \hookrightarrow e_2$ or is a value.

$$\frac{(1+1)+2 : \mathtt{Int}}{(1+1)+2 \hookrightarrow 2+2}$$

The two Steps of Soundness

Progress:

If $e_1 : t$**, then** $e_1 \hookrightarrow e_2$ **or is a value.**

Preservation:

If $e_1 : t$ **and** $e_1 \hookrightarrow e_2$**, then** $e_2 : t$**.**

$$\frac{\begin{array}{c}(1+1)+2 : \mathtt{Int} \\ (1+1)+2 \hookrightarrow 2+2\end{array}}{2+2 : \mathtt{Int}}$$

The two Steps of Soundness

Progress:

If $e_1 : t$**, then** $e_1 \hookrightarrow e_2$ **or is a value.**

Preservation:

If $e_1 : t$ **and** $e_1 \hookrightarrow e_2$**, then** $e_2 : t$**.**

```
(1+1)+2:Int
```

The two Steps of Soundness

Progress:

If $e_1 : t$**, then** $e_1 \hookrightarrow e_2$ **or is a value.**

Preservation:

If $e_1 : t$ **and** $e_1 \hookrightarrow e_2$**, then** $e_2 : t$**.**

$(1+1)+2 : \mathtt{Int} \hookrightarrow 2+2$

by Progress

The two Steps of Soundness

Progress:

If $e_1 : t$**, then** $e_1 \hookrightarrow e_2$ **or is a value.**

Preservation:

If $e_1 : t$ **and** $e_1 \hookrightarrow e_2$**, then** $e_2 : t$**.**

$(1+1)+2 : \mathtt{Int} \hookrightarrow 2+2 : \mathtt{Int}$

by Preservation

The two Steps of Soundness

Progress:

If $e_1 : t$, then $e_1 \hookrightarrow e_2$ or is a value.

Preservation:

If $e_1 : t$ and $e_1 \hookrightarrow e_2$, then $e_2 : t$.

$(1+1)+2 : \mathtt{Int} \hookrightarrow 2+2 : \mathtt{Int} \hookrightarrow 4$

by Progress

The two Steps of Soundness

Progress:

If $e_1 : t$**, then** $e_1 \hookrightarrow e_2$ **or is a value.**

Preservation:

If $e_1 : t$ **and** $e_1 \hookrightarrow e_2$**, then** $e_2 : t$**.**

$(1+1)+2 : \mathtt{Int} \hookrightarrow 2+2 : \mathtt{Int} \hookrightarrow 4 : \mathtt{Int}$

by Preservation

The two Steps of Soundness

Progress:

If $e_1 : t$, then $e_1 \hookrightarrow e_2$ or is a value.

Preservation:

If $e_1 : t$ and $e_1 \hookrightarrow e_2$, then $e_2 : t$.

$$e : t \hookrightarrow e_1 : t \hookrightarrow \ldots \hookrightarrow e_i : t$$

If e_i is a value, we are done!

Since e_i has a type, $e_i \neq \mathtt{crash}$!

The two Steps of Soundness

Progress:

If $e_1 : t$**, then** $e_1 \hookrightarrow e_2$ **or is a value.**

Preservation:

If $e_1 : t$ **and** $e_1 \hookrightarrow e_2$**, then** $e_2 : t$**.**

Soundness:

If $e : t$**, then** $e \not\hookrightarrow^{*}$ crash**.**

The 3 Steps of Programming & Types!

Step 1: Coding

Full of errors!

Step 2: Compilation

Few times & fast.

If $e : t$,

Step 3: Execution

Many times & takes long.

then $e \not\hookrightarrow^{*} \mathtt{crash}$.

Thanks!