KONIG

Final Presentation

Lord Crawford
Matteo Sandrin
Delilah Beverly
The Team

Lord Crawford
SEAS '22 Computer Science
Graph theory was my fav part of data structures :)

Matteo Sandrin
SEAS '21 Computer Science
If it's broken, it's not my SEGFAULT

Delilah Beverly
Barnard '22 Computer Science
<span>insert clever CS joke</span>
What is Konig?

GOAL
Making the creation and manipulation of graphs easier and more enjoyable

Overview
- KONIG is graph manipulation programming language
- Named after the "Seven Bridges of Konigsberg"
- It provides strong primitives (Graph, Edge, Node)
- Features a mixture of Java-like and C-like syntax

```java
ko int main() {
    node<string> n1;
    node<string> n2;
    node<string> n3;
    graph<string> g;
    int i;

    // initialize graph
    g = new graph();
    n1 = new node("Matteo");
    n2 = new node("Delilah");
    n3 = new node("Lord");

    // add nodes to graph
    n1 @ g;
    n2 @ g;
    n3 @ g;

    // fully connected graph
    setEdge(g, n1, n2, 1.0);
    setEdge(g, n2, n3, 1.0);
    setEdge(g, n3, n1, 1.0);

    for (i = 0; i < g.nodes.length; i++) {
        printString(g.nodes[i].val);
    }
}```
Konig is a superset of MicroC

Most objects are heap-allocated (like Java)

Java-like object initialization

Java-like composite type syntax

Konig is a superset of MicroC

n = new node{}
Key Features
Graph Features

**Graphs:**
```java
graph<int> g1 = new graph<>();
list<node<int>> = g1.nodes;
list<edge> = g1.edges;
n @ g1; // add n1 to g1
n ! g2; // del n1 from g1
viz(g, "out.pdf");
```

**Edges:**
```java
edge e = getEdge(g, n1, n2);
setEdge(g, n0, n1, 0);
setDirEdge(g, n0, n1, 0);
getEdge(g, n1, n2);
deleteEdge(g, n1, n2);
edge.weight;
edge.directed;
```

**Nodes:**
```java
node<int> n0 = new node<>();
neighbors(g, n1);
n0.val;
```
Graph Visualization

```java
1   int main() {
2       node<string> a;
3       node<string> b;
4       node<string> c;
5       node<string> d;
6       node<string> e;
7       graph<string> g;
8
9       a = new node("A");
10      b = new node("B");
11      c = new node("C");
12      d = new node("D");
13      e = new node("E");
14      g = new graph();
15
16      a @ g;
17      b @ g;
18      c @ g;
19      d @ g;
20      e @ g;
21
22      setEdge(g, a, b, 1.11);
23      setEdge(g, a, c, 2.22);
24      setEdge(g, a, e, 3.33);
25      setEdge(g, e, c, 4.44);
26      setEdge(g, b, d, 5.55);
27      setEdge(g, c, d, 6.66);
28
29      viz(g, "./graph.pdf");
30   }
```
Types

- int
- bool
- string
- float
- list<type>
- void

- edge
- node<type>
- graph<type>
Functions

- Identified by the custom keyword "ko"
- C-style function syntax
- Extensive set of built-in functions

```ko
int add(int x, int y) {
    return x + y;
}
```
## Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operands</th>
<th>Return type</th>
</tr>
</thead>
</table>
| `a @ g`  | `a` is a node  
`g` is a graph | graph |
| `a ! g`  | `a`  
`g` | graph |
| `a + b`  | `a` is an int, float  
`b` is an int, float | int, float |
| `a - b`  | `a`  
`b` | int, float |
| `a / b`  | `a`  
`b` | int, float |
| `a * b`  | `a`  
`b` | int, float |
| `a > b`  | `a` is any type  
`b` is any type | bool |
| `a < b`  | `a` is any type  
`b` is any type | bool |
| `a => b` | `a` and `b` have the same type | bool |
| `a <= b` | `a`  
`b` | bool |
| `a == b` | `a`  
`b` | bool |
| `a and b` | `a` is a bool  
`b` is a bool | bool |
| `a or b`  | `a` is a bool  
`b` is a bool | bool |
| `not a`  | `a` | bool |
Testing

- We built a custom testing script in Python
- We have pretty good coverage over the code base
THANK YOU!

Any questions?