BOOP: How to Right Code!

A novel problem-solving tool for students of introductory computer science. This is designed to promote structured thinking, strong design choices, self-reflection and a well-probed problem-solving stream.

📝 Language Syntax

The ICS language syntax is designed to be simple and intuitive, allowing users to write structured documents easily.

> - **Sections** follow tag-based hierarchy (similar to HTML!).
> - **Main-tags** (intuitively, these are the 'broad' sections) use double angle brackets (`<< >>`).
> - **Sub-tags** (intuitively, these are the sub-sections) use a simple `tag_name:` format

---

🎯 Main Tags & Structure

1. 📌 Problem Block

```
<<problem n: "Title of the Problem"

    ... (all other blocks go here)

problem>>
```

2. 📝 Blueprint

This lays out the correctness criteria for the input and output of the program.

```
<<blueprint

    requires: [conditions that must hold before execution]

    ensures: [conditions that must hold after execution]

blueprint>>
```

3. 🛠️ Operational Steps

This section is for the user to write down 'informal' steps as if they were explaining a human what to do to solve the problem.

```
<<operational steps

    step 1: [description]
    step 2: [description]
    ...
    step n: [description]

operational steps>>
```

4. 💻 Ocaml Code

Write your standard ICS-friendly OCaml code here. It will be validated by the ICS OCaml validator!

```
<<ocaml code

    (* Write your OCaml code here *)
    let rec search arr target = ...

ocaml code>>
```

5. 📖 Proof

Prove that your blueprint correctness criteria holds.

```
<<proof

    <<induction

        base case: [describe the base case]
        induction hypothesis: [assume for i]
        inductive step: [prove for i+1]

    induction>>

    <<invariant

        pre-condition: [before the loop starts]
        after the ith step: [what holds true]
        after the (i+1)th step: [expected result]
        post-condition: [after loop ends]

    invariant>>

proof>>
```

Syntax rules:

1. Each solution must be enclosed in an appropriate problem tag.
2. Each main tag must contain all its sub-tags. If the user does not have anything to write under a tag, they may leave it blank.
3. The header must contain all required information.

Usage:

1. Install dependencies
2. Create your .ics file
3. Open the command palette (Cmd+Shift+P or Ctrl+Shift+P)
4. Type ICS
5. Select ICS: Compile ICS

Information

Acknowledgements

Thank you to Prof Aalok D. Thakkar for his guidance and mentorship.

Version Info

Developer and Maintainer: Vaani Goenka
@ email: vaani.goenka_ug2024@ashoka.edu.in
Changelog
0.0.1
Initial release
Basic syntax highlighting
Auto-completion for keywords
Document validation
HTML compilation