Assignment 1 Slides

Minesweeper Live Stream

Assignment 1

From a practical perspective . . .

- You will write a C program called Minesweeper
- It will all be in a single file called `minesweeper.c`
- Submission is through the `give` system

Sequence of Commands

Commands will always be in a particular sequence:

- First integer is the type of command
- Other integers are the extra information that command needs
- Your program will receive one or more commands
- You will process each command in turn
- After each command has been processed, you will print out the minefield

Submit early, submit often

Using "give" will record your submission and back up your work

- It's much harder to lose your assignment code if we have it!
- If things go bad, you can roll back to previous versions
- You can access your previous versions using our git repository
- The following link is also available in the assignment page:

`https://gitlab.cse.unsw.edu.au/z5555555/20T1-comp1511-ass1_minesweeper/commits/master`

How will your code be tested?

Your program will be run with a series of test cases

- These tests will not be exactly the same as our autotests
- Remember to check all possible inputs you can think of
- Writing your own test files is potentially very useful

Marking

How do you earn marks in this assignment?

- **Close to a pass (40-50%)**
  - A solid attempt at stage one
  - Being able to place some mines
  - Not necessarily dealing with multiple commands
- **Pass (50-64%)**
  - Code runs without errors
  - A serious attempt has been made at the assignment
  - Able to check how many mines are in rows or columns (hopefully both)
  - A higher mark will be given for completion of stage 1 and dealing with multiple commands

Marking Continued

- **Credit (65-74%)**
  - Successfully implements all of Stage 1
  - Some effort on Stage 2 will push marks higher
  - Code is reasonably readable
  - Shows some use of functions
- **Distinction (75-84%)**
  - Successfully implements both Stage 1 and 2
  - Any effort on later stages will award more marks
  - Code is easy to understand and readable
  - Uses functions to separate code for readability

Marking Continued

- **High Distinction (85%+)**
    - Successfully implements Stages 1-3
    - Stage 4 completion will push marks closer to 100%
    - Code is perfectly explained and elegant to read
    - Functions are used extensively to organise code

Free Marks!!!

Yep . . . get them right here!

Make your code understandable and readable!

- Follow the Style Guide
- This means correct indentation and consistent use of bracketing
- Use variable names that are understandable to a reader
- Have clear comments explaining your intentions (even if the code is not functional)
- Structure your code file so that different sections are clear
- Use functions to separate repetitive code

Hall of Fame

Extra Challenges that are worth bonus Marcs (not actual marks)

These are optional!

- Use the sleep() function to do animated explosions?
- Add colours to the output so the board looks more interesting and/or informative
- Create a reveal function that acts exactly like the game itself (recursive)
- Randomise starting mine locations
- Or any other cool ideas you have!

Questions?

Feel free to ask any questions now!

- Help Sessions have been expanded for one on one consultation if you need help with problems
- There's now a Help Session on every day of the week
- Details are on the Course Website