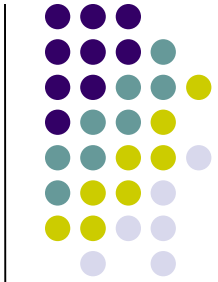
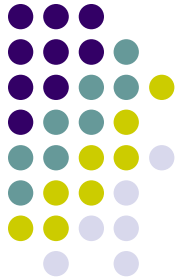


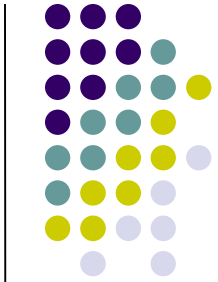
Debugging

Debugging?

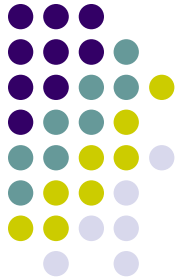


- Bug – problem in the code
- Removing bugs, i.e. problems
- In narrow sense it represents: controlled program execution and insight into program state, in order to fix problems (remove bugs).
- In wider sense is comprises all possible (and impossible) actions in order to detect and fix problems.

Testing



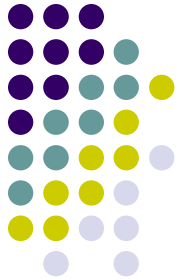
- “Program usually works perfectly, until we run it.”
- Two kinds of checking: verification and validation
- Verification checks whether program does what we want it to do (what we think it oath to do).
- Validation checks whether program solves our problem, i.e. does what we need. (Note that what we want program to do is not necessarily same as what we actually need!)
- Testing does not have to be formalized. But it is very recommended, and common, for more serous projects.



Fixing problems

- Program usually doesn't work until we debug it.
- Test case: input (excitation) – expected output (response)
- Why do we get unexpected output?
- We need better insight into program execution.
- Usually available things:
  - Controlled execution:
    - Step-by-step and break-points
  - View of the program state
    - Monitoring variable and memory values
    - Adding control output (printf, or similar)

Fixing problems



- Unusually available things:
  - Controlled execution:
    - Step-back
    - Conditional break-points
    - Stepping through data flow, instead of execution flow
  - View of the program state
    - Expression evaluation in the current context
- But sometimes we do not have anything mentioned here!
  - Then we do what we can.

Some bug classifications

- Catchable (consistent, reproducible) bugs

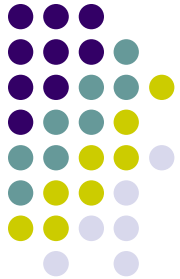
  For the same input we always get the same wrong output

  - Easy to reproduce bugs
    - Input that causes wrong output is easy to create in the controlled environment
  - Hard to reproduce bugs
    - Input that cause wrong output is hard (takes a lot of time and effort) to create in the controlled environment
- Elusive (inconsistent, stochastic) bugs

  For the same input we get wrong output only sometimes

  The cause is existence of unwanted or uncontrolable influence on the program execution, which is not included by the input vector
  Possible reasons:
  - Reading from uninitialized memory
  - Relying on addresses of dynamically allocated memory
  - Triggering undefined behavior, as defined by the language

Digression: undefined states and behaviors

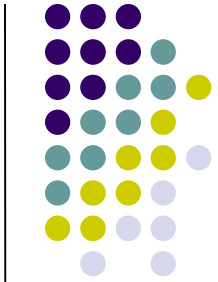


- It means that the behavior can be different between two executions of the same code, on the same system.

Some more prominent examples of undefined behaviors in C language:

- Signed integer overflow
  - Including left shift
  - Only for signed integers! Unsigned integers have very different definition.
- Accessing array element out of boundaries
  - But reading an element form the memory just after an array is OK

Some bug classifications



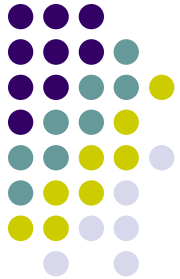
- Simple bugs

  Problem manifestation is tightly connected to bug location in the code.

- Complex bugs

  Problem manifestation is very remotely connected to bug location in the code.

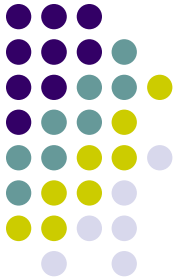
How to debug successfully



- **Understand the system**
- **Make it fail (reproduce the bug)**
- **Quit thinking and look**
- **Divide and conquer**
- Change one thing at a time
- Keep an audit trail (keep notes)
- Check the plug (the problem might be elsewhere)
- Get a fresh view
- If you didn’t fix it, it ain’t fixed

(David j. Agnas, “Debugging”, 2002.)

The most important advice!



Try to solve the problem (and keep trying)