Types of error

1. Data reference error.
2. Data detection error
3. I/O related error
4. Computation related error.
5. Comparision related error
6. Control flow error
7. Interface related error
8. Comments/ ~~Incorrect~~ documentation error
9. Standarisation error

Debugger

1. Static
2. Dynamic

Static Debugging: In this process of ~~b~~ debugging, insert the ~~be~~ debugging codes into the source program and recompile and run. In this case, debugging code to print the suspected variables using the statement for printing in the source program by the programmer himself.

classmate

```
#ifdef DEBUG
    .
    .
    .
#endif
```

```
#include <assert.h>
#define NDEBUG

assert (0 <= i && i <= n)
```

Dynamic or interactive debugging: It executes an executable program so it can be stopped, examined, altered, continued interactively for looking at the variables of the user program.

- → breakpoint setting
- → initiation
- → modification
- → testing
- → Display

Breakpoint debugger: The programmer can define a set of conditional / unconditional breakpoints during using breakpoint debugger.

☆Heisen bug

Meta debugger: It provides facility to debug itself.

Kernel debugger: It can debug kernel program.

Tele debugger:

Debuggerwith same process:
↳ same memory location.

Debugger with separate process: This kind of debugger executes a different process for the debugger and debugging. If necessary, debugger can read and modify debugging memory.
Also it can ~~control~~ access hardware components.

- Unit Test
- Execution sequencing
- Unconditional breakpoint
- Trace
- Single step execution.
- Reverse execution
- Conditional breakpoint
- Watch points
- Checkpoints

classmate

Trace: possible to trace flow of logic at different level.

Single step execution: execute one instruction at a time and check the present state of the program.

Watch points: also breakpoints and are related to the variables used in an expression of the program. During execution of the program, the value of an expression is related with the variable changes, and the break point is triggered for execution.

7/11/19

Input: Users Program
Output: Detection of nature of errors

Steps 1: [Editing]
Steps 2: [Compilation] - ① Code file that includes some special instructions for debugger.
② Debug information file.
Step 3: [Initiate Debugging]
Step 4: [Breakpoints setting] Set a list of breakpoints and their actions.

classmate

Step 5: [Debug table construction] - It contains statements numbers, debug action, etc.

Step 6: [Controlling] - It executes upto breakpoints

Step 7: [Interrupt calls] -

Step 8: [Debug ~~connection~~ conversion] - used for passing the debug commands with breakpoints

Step 9: [Breakpoint returns]

Step 10: [Looping]

Step 11: [Stop]

Breakpoint implementation

Input: Source program, list of breakpoints

Output: Code file.

Step 1: [Compilation of source program]

Step 2: [Insertion of no-operation instruction in the code file]

Step 3: [Symbol table construction]

Step 4: [Replacement of no-op instruct^n]

Step 5: [Generation of software interrupt]

Step 6: [Control transformation]

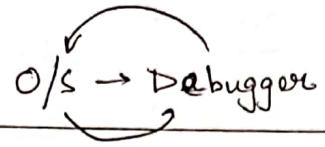
Step 7: [Inspection]

Step 8: [Debug queries]

Step 9: [Continues program execution]

Step 10: [Stop]

classmate



ⓐ via a message passing
ⓑ by a system call
ⓒ by a straight jump (i.e., unconditional)

Debugging using hardware

(gdb) r
(gdb) n
(gdb) p
(gdb) q
(gdb) b
(gdb) d

(gdb) c

Loaders and Linkers.