Debugging

Beatrice Crippa

10/05/2024

Goal: discover and fix issues

- **Unexpected behaviors**: compilation may not fail if we do something illegal but the results are not what we expected;
- **Defensive programming:** anticipate mistakes by making the compilation fail when they happen;
- **Simple debugging techniques:**
  - Print statements;
  - Asserts;
  - Pauses;
  - Range check with Address Sanitizer;
- **Command line debugger (GDB)**;
- **Memory leaks:** Valgrind.

Examples - Debugging

These examples are inspired by `https://github.com/cme212/course` and show basic techniques for debugging as well as an introduction to `gdb`.
See also Defensive Programming and Debugging.

Some useful readings about *undefined behaviour* in `C++`:

- `https://mohitmv.github.io/blog/Cpp-Undefined-Behaviour-101/`
- `https://mohitmv.github.io/blog/Shocking-Undefined-Behaviour-In-Action/`

Exercise 1 - Debugging with gdb

The program `student2` in the directory `01-debug-intro` implements a class `Student` with two attributes:

- Name;
- StudentID.

The student's name is an instance of class `Name`, with two attributes of type `std::string`:

- First name;
- Family name.

The program wants to create a pointer to a new student, print the name and studentID and finally delete it.

There is a run-time error leading to segmentation fault.

Using `gdb` find the issue and fix it.

Exercise 2 - Debugging (advanced)

The program `integer-list` in the directory `02-bug` has:

- a compile error;
- a run-time error;
- a memory leak;
- a possible memory leak that is not captured by the `main`.

Find all the issues and fix them.

Get help by using `gdb` and `valgrind`.

The directory `02-bug-solution` contains the fixed code, please don't look at it before trying to solve the exercise by yourself!