DATA STRUCTURES AND ALGORITHMS

Lab 02 – Practical Programming

Dr. Faisal Shafait (faisal.shafait@seecs.edu.pk)
Consultation Hours: Mon 3pm – 4pm (CR 01)

TA (quizzes, assignments): None
Lab Engineer (labs): Maryam Sajjad (maryam.sajjad@seecs.edu.pk)

Agenda

- Version Control
- Debugging
- Random Programs
- Unit Testing

Version Control Systems

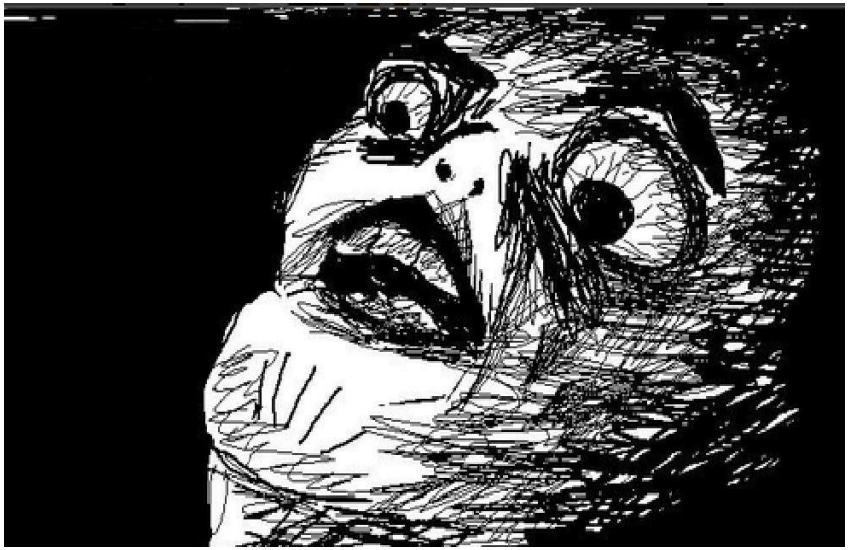
- what are they?
- how are they used?
- features of version control
- a short demo of git

Dealing with Change

- How do you manage your coursework?
  - Modifying existing code (using Q1 for a basis for Q2)
  - Backing up working code
  - Checking if an idea works
  - Sharing code in group projects

(Bad) Solutions

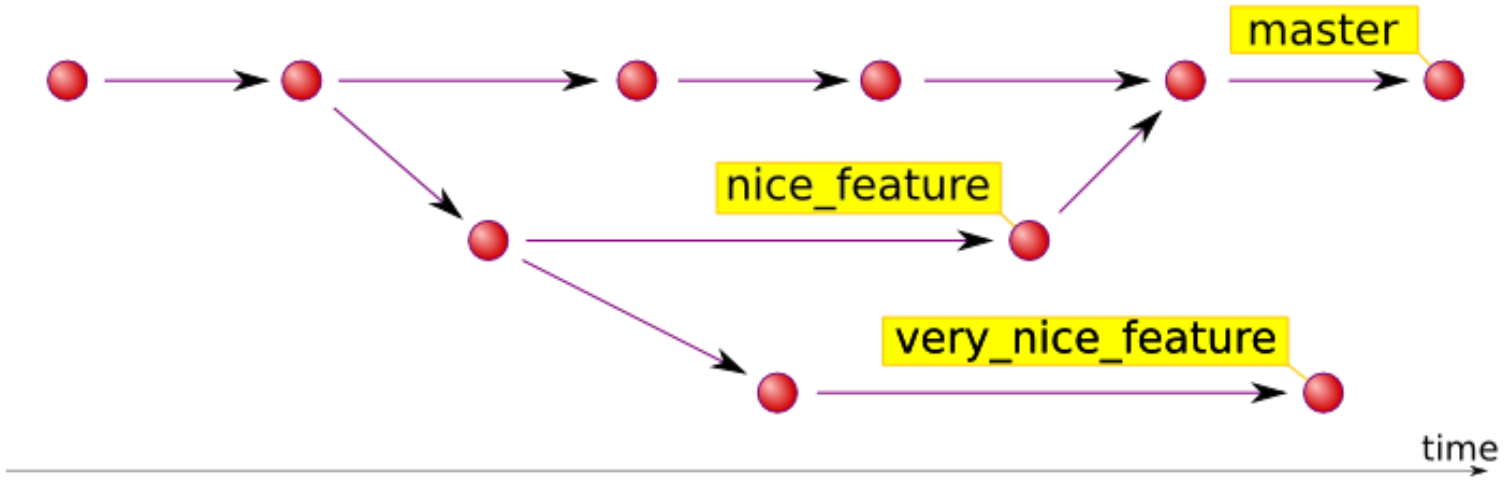
- Copying
  - assignment_working.cpp
  - assignment_old.cpp
  - assignment_new.cpp
  - assignment_new2.cpp
  - assignment_tmp.cpp)
- Copy entire directories
- Emailing code to people



Version Control

- Files are kept in a repository
- Repositories can be local or remote to the user
- The user edits a copy called the working copy
- Changes are committed to the repository when the user is finished making changes
- Other people can then access the repository to get the new code
- Can also be used to manage files when working across multiple computers

Code Evolution in Version Control



Version Control in Practice

- https://github.com/
- https://bitbucket.org/
- Typical operations
  - clone
  - checkout
  - add
  - commit
  - push

Debugging

- Visual Studio Editor
  - Set break-point
    - Click on the left of a line of code to set break-point there
  - Run program line by line
    - Step Over (F10): Do not dig into a function call, just run the function and return the result
    - Step Into (F11): Go into the function code and debug it line by line
  - Inspect intermediate values of variables
    - Move the mouse pointer over the variable

Randomizing Programs

- Generating random numbers
  - `void srand(long seed)  // set the seed`
  - `long rand()  // generate a random number`
    `// in the range [0, RAND_MAX]`

Sample usage of `rand()`

```
#include <XYZ> // figure out yourself

int main () {
    // call just once, at the very start
    srand( time( NULL ) );
    for(int i=0; i<10; i++){
        cout << rand() << '\n';
    }
}
```

Testing

- Testing is the backbone of programming
- Each function you write needs to be tested
- Test-driven development
  - First write the test cases, then the code
- Over 80% code in large programs is just for testing

Lab Task 2.1

- Write a program that simulates the throw of a dice.
  - The output sequence may look like 4 1 6 3 2 ..

- LMS Upload:
  - Source Code of the program

Lab Task 2.2

- Consider the moveMin problem discussed in the class – all elements in the array are sorted except the last element, e.g.

  3, 5, 12, 24, 25, 27, 15

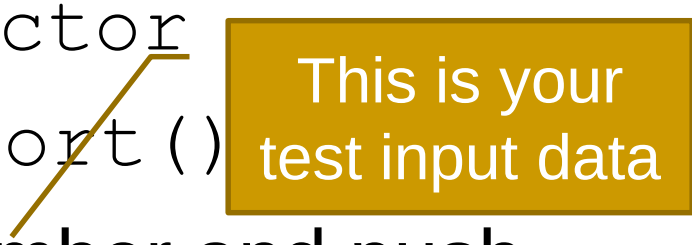
- Implement the naïve solution using two nested loops

```
bool moveMin(vector<int> &in, vector<int> & out)
```

- Write test cases

Lab Task 2.2 – Test Cases

- Write a function `bool testMoveMin()`
- It should generate a random array of integers in the range 1-100.
- Store the array in a `std::vector`
- Sort the vector using `stl::sort()`
- Generate another random number and push it at the end of the array



Lab Task 2.2 – Test Cases

- How do you create the test output data?
  - Copy the test vector into another vector and sort the new vector
- Run the test case
  - Call the implemented function with the test input data and compare its result with the test output data

Lab Task 2.3

- Now implement the moveMin method using the single `for` loop
- Run the test cases to verify the correctness of your newly implemented faster algorithm
- Generate test cases of different sizes (10, 100, 1000, 10000, 100000, …) and note the difference in running times of both algorithms
  - Ask Google how to calculate running time of your C++ function / code

Lab Task 2.3

- Now run the test case with 10000 input dimension at least one hundred times and compute
  - Best case running time
  - Worst case running time
  - Average case running time
- LMS Upload:
  - source code
  - Short report about the running time comparison of the faster and slower algorithms as well as the best / worst / average running times

Lab Task 2.4

- Create an account on bitbucket.org with you seecs email address
- Create a repository called cs250-lab2
- Add the code of Lab Task 2.2 to the repository
- Make a clone of the repository
- Modify the code in the clone to achieve Lab task 2.3 and commit the code
- Add comments to the lab task 2.3 and commit again
- Push the modified code to the bitbucket repository
- LMS Upload: a screen-shot of the commit history of cs250-lab2 on bitbucket