Programming Techniques project, Spring 2011

Alex Muscar

March 2, 2011

Professor:	Costin Badica
Teaching assistant:	Alex Muscar
Location:	S7a
Time:	Second and fourth Wednesday of each month, 14:00 - 18:00
Attendance:	not mandatory but recommended

Problem list

You can find a list of proposed problems here

Deliverables

The project should contain the documentation in both printed and electronic format (PDF) and the complete source for the application. The documentation must be typeset using LaTeX.

Note: I/O uses standard input and output *stdin* and *stdout*.

Project documentation

1. Problem statement
2. Pseudocode for the main algorithm used; the pseudocode should use the conventions defined in the lectures or those in *Introduction to Algorithms*
3. A detailed description of the application containing:
   - a high level architectural overview of the application
   - a specification of the input
   - a specification of the output

- a list of all the modules in the application and their description
- a list of all the functions in the application, grouped by module; for every function the following details must be provided:
  - a description of what the function does;
  - a description of every parameter; and
  - the meaning of the return value.

4. At least 5 *non trivial* input samples and their corresponding outputs
5. The complete source of the application (*.c*, *.h* and *Makefile* files)

Grading

While grading the following will be taken into account:

1. The project must successfully compile and build; when using GCC the build process should use the *make* tool and a *Makefile*
2. The code must follow the C99 standard (e.g. no compiler-specific extensions)
3. The code must be portable; the project must compile under two different compilers (e.g. Visual C++ and GCC)
4. The code must use the Allman indent style and obey at least the following rules:
   - blocks must be indented using 4 spaces (no tabs)
   - every control construct (e.g. *if*, *else*, *while*, *for*) must be braced — even those with single-line bodies
   - the variable names should be descriptive, but not verbose
   - the use of global variables should be kept to a minimum
   - the use of preprocessor macros should be kept to a minimum
5. The code should be commented (every function and important variables and code blocks inside functions)
6. The code must be modular (minimum 2 *.c* files and minimum one *.h* file)
7. The use of an automated method for generating test data

Short setup guide

You will need to install at least Visual C++ Express, MinGW and MikTeX.

The Visual C++ install is straightforward: download the installer and once it's done you have and IDE, a compiler and a debugger.

On the other hand setting up a development environment for GCC is more involved. You'll first need to install MinGW. Follow the instructions under *Graphical User Interface Installer* from *The MinGW Getting Started Guide*. Make sure you select *MSYS* for installation. Also make sure you set the appropriate environment variables.

Next you'll probably want an IDE for developing projects under GCC. Both NetBeans and Eclipse have good plugins for C/C++ development. Choose whichever you like and follow the installation instructions. They should both be easy to setup.

Next download MikTeX. Use the net installer to spare some bandwidth and download time. After you have MikTeX download the Latex Editor and install it.

If you have any issues you can reach me via email or during the project lab hours or (if you're lucky) in room S2.

Happy hacking.

Resources

Visual C++ Express

NetBeans

Eclipse CDT

MinGW

The MinGW Getting Started Guide

MSDN

MikTeX

Latex Editor

Bibliography

1. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009). *Introduction to Algorithms* (3rd ed.). MIT Press. ISBN 0–262–03384–4.

2. Kernighan, Brian W.; Ritchie, Dennis M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall, Inc. ISBN 0–13–110362–8.