Computer Programming using C
Lecture 9:
Practical software design and your assignment

Prof. Stephen Smith
E-mail: stephen.smith@york.ac.uk

Based on lecture notes by Dr Julian Miller

Program File Structure

- It is useful to build program into a collection of files that handle different aspects
- The minimum number of files is three
  - .c file containing the main function
  - .c file containing all functions
  - .h file containing all defined constants and function prototypes

Software Development Model

1. Requirements
2. Analysis
3. Specification
4. Design
5. Implementation
6. Verification and testing
7. User Manual

Requirements (2 marks)

- State the requirements you have been given
  - Explicit: the ones clearly stated in the assignment requirements
  - Implicit: the ones that are implied by the assignment
    - E.g. in C, Operating System, Graphics/midi libraries

Analysis (10 marks)

- Aim: to understand at a high level what the program needs to do
  - Treat the program as a *black-box*, so don’t talk about program elements inside the box
- User inputs to the box
  - Type of data input (key pressed expected, numbers/strings from keyboard)
  - range of data
- Outputs
  - Graphics, music, text to screen
  - Ranges (pixels, note pitch values, instruments…)
- Analysis of processes and relevant mathematical formulas
  - e.g. Newtonian laws of bodies falling under gravity
  - Methods of changing key, musical theory
- Identify who will use the program
  - What knowledge are they assumed to posses?

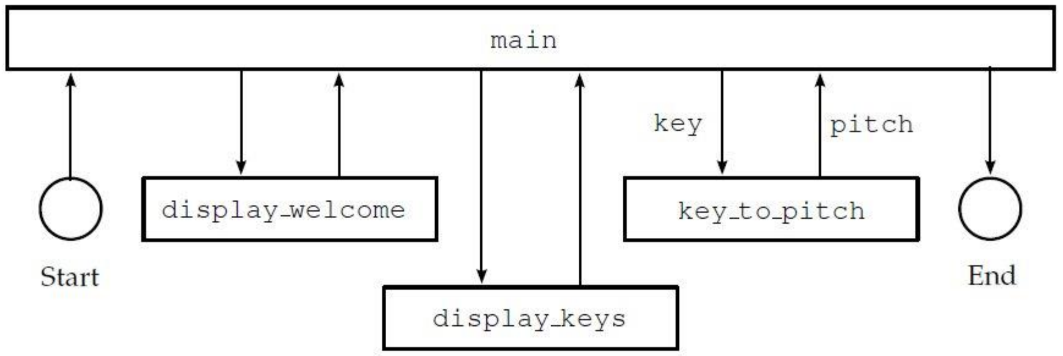
Specification (3 marks)

- Formal specification of what the program must do
  - Expands on requirements using analysis
    - E.g. draws a movable stick-man on the screen, able to launch a drawn projectile towards a target
    - Plays music using a number of user chosen instruments in a variety of keys and styles etc…
  - Explain what the program must do and what is optional
- Do not talk about how the program is implemented
  - E.g. functions needed

Design (20 marks)

- **User interface design**
  - What information is requested from the user and when
  - Explain in detail what will be drawn on the screen (pixel ranges) and in what sequence and where or what notes will be played (pitch ranges)
  - Describe what input from the user is expected and acceptable
  - Explain how you will validate user input
- **Structure of program in terms of functions**
  - Draw a structure diagram showing the data flow between functions
  - List functions that are required
  - What data will they take, what will they do?
  - Give a pseudo code description of all algorithms used
- **Data Tables**
  - Specify a data table showing the name, type and value range of each variable you will use.
- **Logical Flow**
  - There should be a logical flow through your design, starting from the specifications and ending up at a full design for the program.
  - Refer to your specification

Structure Diagram and Function Table



Function	Arguments	Returns	Description
main	None	Default value (integer)	Driving function, calls all others, always returns 0
display_welcome	None	None	Displays the welcome message to the user
display_keys	None	None	Displays the instructions for how to use the keyboard to the user
key_to_pitch	Key code (integer)	Pitch of note (integer) 0 if the key not valid	Converts a key number to a pitch, or to zero if the key wasn’t valid,

Pseudo Code Descriptions

```
function main
   call display_welcome function
   call display_keys function
   begin loop
         call getch to get a key from the user, put the result in key
         if key is zero then
            call getch again, putting the result in key
         end if
        call key_to_pitch passing key, assigning the result to pitch
        if pitch is not equal to PITCH_INVALID
           turn on a midi note of pitch, on channel 1 with velocity 64
           pause for NOTE_DURATION
           turn off a midi note of pitch, on channel 1
        end if
   loop while leave is equal to 0
end function main
```

Implementation: Report (3 marks)

- Give a table showing the source files and what they contain (functions, prototypes, global constants…)
- Document and justify design changes
  - Try not to make too many of these

Implementation: Code (22 marks)

- This should be nicely formatted and have consistent indentation
- It should use #defines
  - E.g. global constants, preprocessor directives
- It should use source code files
  - Headers, multiple source code etc
- It should be well documented
  - Comments
  - Variable names
  - Function names
  - Lots of smallish functions
- Appropriate use of structures and pointers
- Mouse, keyboard and sound handling
- Marks are deducted for explicit numerical values buried in the code and global variables

Testing and Verification (5 marks)

- Test strategy
  - How you designed your program so that it works correctly under all conditions
  - Explain why it is *sufficiently* complete
- Test input-output data
  - include all test input data and test results
  - comment on your test coverage
- Modifications
  - detail any modifications you made following test failure and show the results of re-testing.

User Manual (7 marks)

Identify the intended user

  - you identified in your analysis
  - Write the manual with this user in mind

Your user manual should include:

- Installation instructions
  - this may just be copying the executable file for your program;
- System requirements
  - what type of computer is necessary to run the program
  - what kind of expertise you are expecting the user to have
- Usage Instructions
  - explain all your programs features and how to use them
  - ideally with examples;
- Frequently-asked questions (FAQs)

Maturity, Consistency, Presentation and Innovation (13 marks)

- Is the program reasonably concise and efficient (2)
- Is it mature (2)
- Report Presentation (2)
- Innovation and sophistication(7)

Demonstration (15 marks)

- Does it meet the specification?
- If not what aspects are missing?
- Is it innovative?
- Does it work well?

Summary

- Please follow the assignment requirements carefully
- Marks are awarded for the design and documentation of the program
  - as well as the code itself
- Ensure you submit your assignment in good time