Lab 1 — A Simple C Program

Introduction

This lab requires you to write a simple C program to familiarize yourself with the lab computer and the C development software.

Program Requirements

Write a C function, `void psquare(int n)`, that prints a square on the screen. The square is to be built by printing asterisks `'*'` and spaces `' '` using the built-in function `putchar()`. The dimension of the sides of the square is passed to the function as its one and only integer argument. For example, `psquare(3)` would print:

```
***
* *
***
```

while `psquare(5)` would print:

```
*****
*   *
*   *
*   *
*****
```

Write a C program that reads one numeric character from the keyboard, converts it to an integer (a value from 0 to 9) and calls the `psquare()` function to print a square of that size on the screen.

If the character is not a digit your program should print only an error message.

Use the `getche()` function to read the character, the `putchar()` function to print the characters making up the square and the `printf()` function for printing error messages. You should put the lines `#include <conio.h>` and `#include <stdio.h>` in your program in order to use these functions.

The `printf()` function can be used with only one argument as follows: `printf ( "Error ..." ) ;`. Use the character `'\n'` (“newline”) as the `putchar()` argument to start a new line in the output. Start a new line immediately after reading the character from the keyboard.

Using Turbo C

You will be assigned an account for the computers in the APSC 380 lab. Type ”login” and enter your user name at the prompt. The first time you log in you should set (or change) your password by using the ’setpass’ command.

Type the command ”TC” to start the Turbo C compiler.

Press alt-E to switch to the edit window and enter your program. When you are done editing, press F2 to save it.

Press F9 to compile the program.

If your program contains syntax (“grammatical”) errors, then error messages will be displayed in the Message window. You can move up and down through the messages and the location of the error will be highlighted in the edit window. While in the Message window you can press F1 to get more information about an error message.

Type alt-E to switch back to the edit window, fix the error(s) and repeat the edit/compile process until your program compiles without errors.

You can press the F1 function key at any time to access the help menu.

To get documentation for any function (such as `getche()`), put the cursor over the function name (or other language item) and press control-F1.

When your program has compiled properly, type control-F9 to run it. When the program terminates you will be returned to the edit window; type alt-F5 to switch back to the run-time output window to check your output.

Hints

The ASCII codes for the characters from '0' to '9' have values ranging sequentially from 48 to 57 ('0' is 48, '1' is 49, etc). The fact that the ASCII character codes for digits are sequential and in increasing order makes it easy to convert from the character value to the number represented by that character.

As in any situation where you are asked to write a program, you should do the following:

- try to break this problem down into a sequence of simpler steps.
- continue the process of decomposition until you think you can solve the problem using C language constructs that you are familiar with
- verify the correctness of each step or part of your solution before testing the complete program
- if you run into problems, try to figure out the cause and try an alternate approach before asking for help
- try to predict what should and would happen with extreme input values (e.g. negative values, zero, large positive values).

Demonstration and Lab Report

After your program is running correctly use the DOS `PRINT` command to print your program on the laser printer in the lab. If you are asked for the name of the print device, enter `LPT2`.

After you have printed out the program, demonstrate its operation to the TA. He will ask you some simple questions about the operation of your program to make sure you understand your work.

For your write-up for this lab you only need to hand in the commented listing of your program. Comments should include the course and lab number, the file name, your name, your student number and the date. A comment before each function should contain a brief (3 sentence or less) description of the purpose of the function, the purpose of each argument and the meaning of the return value. Within each function put short (1-sentence) comments explaining the *purpose* of each major section within each function. Your comments must be correct.

Program Style

Marks will be deducted for flagrantly bad style such as mismatched or inconsistent indentation, "cute" but meaningless variable names (such as `betty, fido, etc.`), or comments that are incomplete or factually wrong. Use the K&R indentation style used in the notes if you are not already familiar with another style.