Chapter 3

Let's Program in C

Translated by Spectras ; Many thanks to him/her! - Website: http://www.tiwiki.org/

I: International C standards

If you would like to know more about general C programming, you should go buy a book. You should have a look at a few of the available books and choose whichever pleases you the most. You should not make the size of the book a point, since small ones often drop some important points, while big ones tend to have much more in them that one usually needs, which may bore you in the long run.

Most C programming books are focused on the PC environment, but you should remember that C was made to be as portable as possible, so the same language can be used on all programmable devices. The other way around, a lot of what you will learn in this tutorial will apply on other platforms as well. However, avoid books about windowed applications, since those are much more complicated than text-based programs, and are not part of the C standard anyway.

Finally, the definite reference for the C language is The C Programming Language, 2nd edition, by Brian W. Kernighan and Denis M. Ritchie, the very designers of the C language. This book is also often referred to as *K&R*, from the authors'initials. Please note, however, that as a reference, it may not fulfill the needs of a beginner.

One should know that the C standard allows compilers to add extensions to it, and, actually, most compilers do it. This has a lot of advantages, since it allows specific solutions to specific problems that were not expected at the time of the writing of the C standard. On the other hand, programs using compiler extensions will not work with other compilers, thus defeating one of the key features of the C specification : generic and portable code. As TIGCC is the only full-grown compiler for TI calculators, and some things just cannot be done without them, we will use TIGCC extensions every now and then in this tutorial.

II:\ About TI calculators

Although the language itself is universal, the environment that will host your programs is not, and one must know at least a few important facts about it to do anything a bit sophisticated. This will therefore be a quick review of the key features of TI calculators.

There are three calculators models based on the Motorola 68k processor: 89, 92 and V200. Although they look quite different, the electronics is almost the same, except for the larger screen on the 92/V200 and the reduced keyboard of the TI-89. The 89 itself comes in three hardware versions, commonly referred to as HW1, HW2 and Titanium. Those calculators are sold with TI's operating system, the Advanced Mathematical Software (AMS). This software handles all a TI calculator offers out of the box: from displaying things on the screen to solving equations and drawing graphs. There are a lot of different versions, and new ones a released on a regular basis. Written in the permanent part of the memory, this software is sometimes called ROM or ROM image. However powerful it may be, AMS is often an annoyance when it comes to creating additional software for TI calculators. One's program has to interact with it, and try to keep compatibility with all past, present and not-yet-released versions so that all users are able to use it.

III:\ A first program

We will begin with a minimal C program that does nothing, and explain what some basic elements of to mean. Here is the source:

```
#include <tigcclib.h>

// Program entry point
void _main(void)
{
    // Place your code here.
}
```

Exemple Complet

A: Comments

Comments are the parts of the source in green. Pure C comments begin with the /* delimiter and end at the next */, which can be many lines ahead. TIGCC also supports C++ style comments, which begin with the // delimiter and extend to the end of the line. Comments are just ignored by the compiler, and are usually used to explain tricky parts of the source, detail variable uses, etc. It is strongly advised to comment your code thoroughly, for it gets very difficult to remember the hows and abouts of each and every pieces of code in a 10000-line project. It gets even more important to have clear and understandable comments in your code when you want to distribute it, work with other people or share it in any other way.

Be careful not to do the opposite mistake either : overloading your programs with useless comments will just confuse people. What exactly make a well-commented program is not agreed upon, but at least, comments should be clear, concise and helpful. That is, commenting that a+b computes the sum of a and b is pointless. Another common error is to not update comments when the code changes, thus leading to mismatched code and comments. And as Norm Scryer said, 'If the code and the comments disagree, then both are probably wrong'.

B: The _main function

The _main function is a special function, in that it is the one that is called when the user launches the program. Though functions will be explainer later, you should know that they are independent pieces of code that can call each other to do specific sequences of actions. The special function _main is defined as "void _main()", which tells the compiler that _main is a function that take no parameters and returns nothing to the caller. The body of the functions follows, enclosed in {}. In this example, it has an empty body therefore it does nothing.

You should know that you can define your own functions, and you can use functions created by other people, using special compilation techniques. Also, lots of AMS features can be used by calling AMS functions, which will be explained in this tutorial as we need them. One defines his own functions thusly:

```
return_type name(argument_list)
{
  // body
}
```

The following chapter will focus more on how to use functions once they have been defined, in particular AMS functions (aka romcalls).