More C

A look into some theoretical concepts

Those of you who tried to run the previous program on **TurboC/C++** (I know there are only a few of you), might have encountered a problem. You just saved and ran the program, but as you pressed Ctrl+F9 on your keyboard or ran it from the menu, it just sat there and did nothing. What happened? I am telling you it ran. How am I so sure? I know my code was correct. So what? Maybe you typed it incorrectly? In that case, you would have encountered some form of error. But if you did not encounter an error, then I am telling you that code ran. Then, why didn't you see it? Actually, you were not fast enough. No human is. Admit it, we, humans, are not faster than computers. Actually, the output just showed up and vanished just like that. So, how to fix that? I am going to tell you that and a lot more in this tutorial. On the other hand, those of you who compiled the code in GCC must have seen it work just fine. Now just take a look at the following piece of code.

```c
#include <stdio.h>
#include <conio.h>

int main() {
	clrscr();
	int radius;
	float area;

	radius = 5;
	area = 3.14*radius*radius;

	printf("Area of the circle is %f\n", area);
	getch();
	return 0;
}
```

Woah! So many new things. Let's digest it slowly over a few tutorials. Firstly, I need to fix the problem of the output not showing up. How to do that? For that, you need to include a header file called **conio.h** *(line 2)*. And on the line previous to return statement, write a call to the `getch()` function residing in **conio.h** *(line 13)*. What does it do? It would tell the output screen, "Hey there! Where do you think you are going? Hold it there man!". And the output screen freezes up right there. It would be interesting if they could converse like that, ehh? I will tell you what happens actually. `getch()` means get character. Its purpose is to read a single byte character from the input. Freezing up the output screen is actually a side-effect of its actual purpose. It makes the program wait until we enter some character. As long as we don’t type anything from the keyboard, it waits and voila! We can see the output. **Conio.h** contains another useful function `clrscr()`, whose job is to clear up the output screen whenever it is called. Keep it in your program so that things don’t get messy when you run a program multiple times. Having known about that, let’s see what C allows us to type in a program.

The C character set

A character is obviously any alphabet, digit or special symbol used to represent any information. C allows the following set of characters.

Alphabets	A, B, C, ..., X, Y, Z a, b, c, ..., x, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
Special Symbols	~ ` ! @ # $ % ^ & * ( ) _ - + = | \ { } [ ] : ; “ ‘ < > , . ? /

It is time we start breaking down the above program and slowly begin to understand what is what and how it works. For that we need to know what are variables, data types and tokens (Oh no! Some more definitions!).

Variables and Data Types

From nursery we grew up learning mathematics. At first we were adding constant numbers (or subtracting or multiplying for that matter) and we were happy. But then came algebra (oh no)! In algebra we encountered x,y and what not. These were variables. These exist even in C. We use variables in C to be more dynamic (I mean come on, dynamic is the new sexy!). Suppose we want the user to enter some value and we need to store it. What do we do? We use variables to hold the value. If we wish to calculate a value and store it, we use variables. For example, in the piece of code at the beginning of this tutorial, the variables are radius and area. So how do the variables store the data? I’ll explain that in the next tutorial. For now, we need to know another super important thing that is the rules governing the naming of a variable. These are:

1. **The length of a variable name may be any characters long but some old compilers only care about the first 31 characters.** I don’t know who would want to do that, but if any of you want to name your variables more than 31 characters just make sure the **first 31 characters are not same. Modern compilers like GCC have no such limit.**

2.**Variable name can consist of only alphabets (both lower or upper case), digits or underscore _.**

3.**Variable name must start with an alphabet or an underscore.**

4.**No commas or blanks are allowed within a variable name.**

5.**No special symbol should be used other than underscore.**

6.**A variable name cannot be a keyword** (what are keywords, read on to find out).

7.This one is not a rule, but I would like to mention that though creating unnecessarily long variable names is a waste of effort, but as a good programming practice, **variable names should be such that when you re-visit your old code or someone else sees it, they should understand what each variable is there for.**

Variables can be used to store different types of data. That maybe simple or complex. What separates one from the other is the data type. In C we need to declare the data type for the variable we are going to use. C supports four different data types.

1. **Basic data types:** These are the data types on which arithmetic can be performed. C has 3 basic data types:

    •Integer type
    •Floating-point type
    •Character type

    In addition to that, we have 5 modifiers: long, short, signed, unsigned and long long. Modifiers are used to change the basic properties of these basic data types when required.

2. **Enumerated data type:** They are again arithmetic types and they are used to define variables that can only attain certain discrete integer values throughout the program.

3. **Void data types:** Type specifier void indicates that no value is available. So what do we use it for! All in good time.

4. **Derived data types:** They are derived using the basic data types. Derived types include: array, structure, union, function and pointer.

Tokens or Lexical Units

The smallest unit of a C program is called a token or a lexical unit (so, we are really breaking down the program, huh!). Tokens in C can be of 6 types:

1. **Keywords:** C keywords are the words that convey a special meaning to the C compiler, that is, the compiler already knows what these words mean. As such, these words can’t be used to name any variables. There are 32 keywords in C which are listed as follows.

auto	break	case	const	char
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

2. **Identifiers:** Identifiers are used as the general terminology for the names of variables belonging to all data types. These are user defined names consisting of arbitrarily long sequence of letters and digits following the rules mentioned above for naming a variable. For example: var_name, _age, gender, new12, etc.

3. **Constants:** C constants refers to the data items that do not change their value during the program execution. Several types of C constants that are allowed in C are:

    •Integer Constants

        •Decimal Integer Constants

        •Octal Integer Constants

        •Hexadecimal Integer Constants

    •Real Constants

    •Character Constants

    •Escape Sequences or Backslash Character Constants

    •String Constants

4. **Strings:** Strings are sequences of characters enclosed within double quotes. A string is automatically terminated with a null character by the compiler. For example: “hello”, “new”, etc.

5. **Special Symbols:** They hold special meaning for the compiler and hence cannot be used for any other purpose. For example: (,), {, }, [,], etc.

6. **Operators:** C operators are symbols that triggers an action when applied to C variables and other objects. The data items on which operators act upon are called operands. For example: +, -, *, etc.

Everything will be discussed in great detail, you just have to bear with me. For now I would like to clarify tokens furthermore by classifying the tokens used in our program above. They maybe classified as follows:

Keywords: int float return

Identifiers: main clrscr radius area

Constants: 5 3.14 \n 0

Strings: “Area of the circle is”

Special Symbols: ({ })

Operators: * =

I hope this clears everything about the different types of tokens. In further tutorials we’ll take up basic data types alongwith modifiers one-by-one and then we’ll look into constants and from there we’ll see where it goes.