My notes from K.N. King's "C Programming A Modern Approach" 2nd version

Piotr Marendowski

March 2023

Chapter 1

Note

In this material I will go over everything from book, trying to summarize every note-worthy subject. I will do it, while learning Latex, so good luck to me.

Contents

Chapter 2

C Fundamentals

2.1 Steps of Executing a C Program

Automated proccess:

1. **Preprocessing** - Preprocessor is executing directives (they begin with #).
2. **Compiling** - Compiler translates program into machine instructions (object code).
3. **Linking** - Linker combines object code and code needed for execution of the program.

2.2 The Genereral Form of a Simple Program

Simple C programs have this form:

directives

```
int main(void)
{
    statements
}
```

Directives - Begin with '#' symbol, they state what headers include to program.
Functions - They are segments of code that take arguments, and returns (or not) a value. Only `main` function is required.

Statements - Commands to execute, mostly end with semicolon.

String literal - Series of characters enclosed in double quotation marks, e.g. `"Hello world!"`.
New-line character - `\n` is an escape sequence, which advances to the next line of output.
Comments - Are ommited in program execution, can be used to comment single line e.g. `/* Comment */`, or block of lines. From C99 we can use one line comments e.g. `// Comment`.

2.3 Variables and Assigments

Variable - Place to store calculation's output, for using in future. Variable's characteristics:

- **Types** - For now, there are two types of variables:
  - `int` - Integer types, can store quite big whole number, but that depends on your computer's architecture.
  - `float` - Can store bigger numbers, as well as digits after decimal point.
- **Declarations** - To use a variable, we first need to declare it. It means that we need to specify variable's type, and name. We can chain declarations with the same type e.g. `int i, sum, x;`. In C99 they can now be declared after statements, not like in C89.
- **Assignment** - We assign value to a variable. Variable is on the left side, while value, expression, formula etc. is on the right side. To assign something to a variable, we first need to declare it. Examples:

```
int i;
float f;
i = 1;
f = 1.5;
```

Initialization

At the default most variables are uninitialized, which means that they have some random - garbage value assigned to them, if we didn't. In declaration we can assign value to a variable, making it an **initializer**, e.g. `int i = 0;`.

2.4 Reading Input

For reading input we need to use `scanf` function, which needs a format string and value to read, e.g. `scanf("%d", &i);`.

2.5 Defining Names for Constants

To define a constant, we need to use a **macro definition**, which is interpreted by the preprocessor e.g. `#define WIDTH 20`.

2.6 Identifiers

Names in C are called **identifiers**. They can begin with the lower-case or upper-case letters or underscores e.g. `times10 my_var _done`. They cannot begin with a number e.g. `10times`. They cannot contain minus signs e.g. `my-var`.

Keywords

There are number of keywords, which are prohibited from using as identifiers.

2.7 Layout of the C Program

We can slice C statements into **tokens**:

```
printf   (   "Height:  \%d\n"   ,   height   )   ;
  1      2       3        4      5      6      7   8
```

Tokens 1 and 2 are identifiers, token 3 is a string literal and tokens 2, 4, 6, and 7 are punctuation.
In most cases we can put many spaces between them. But we cannot put spaces within tokens e.g. `fl oat f;`.