C Basics

Summary for C tutorial sessions

Marcus Birkenkrahe

April 7, 2025

How should I approach a programming assignment?

1. Understand the problem ("What")

   Example: "Write a C program that defines a constant for the speed of light (299792458 meters per second) and asks the user to input a time in seconds. Calculate and print the distance light travels (speed × time) as an integer."

   This exercise asks you to:

   (a) Write a C program (.c)
   (b) Define a constant (integer) value
   (c) Get an integer value as input
   (d) Compute something

2. Plan the solution ("How")

   For the example:

   (a) Open Onecompiler.com/c for a `main.c` template
   (b) Use `#define` for the constant
   (c) Use `scanf` to get input
   (d) Compute by multiplying (`*`) and saving the result (`=`)

3. Implement the solution:

   (a) Write the pseudocode
   (b) Write the C source code accordingly
   (c) Test the program
   (d) Fix according to the debugger messages

What's a C program exactly?

- A C program consists of several parts:

```
*----------------------------------*
| Program header:                  |
| Purpose, author, input/output    |
*----------------------------------*
| Preprocessing directives:        |
| #include [header file]           |
| #define [constant]               |
*----------------------------------*
| main program                     |
*----------------------------------*
```

- The `main` program consists of several parts:

```
int main (void)
{
  // Variable declarations

  // Input from the keyboard

  // Computations with variables

  // Output results

  return 0;
}
```

`printf` and `scanf`

- The general format of `printf` is:

```
printf(" [formatting] ", [arguments]);
```

- Formatting could be `%d` for an integer argument:

```
printf("An integer: %d", 100);
```

```
An integer: 100
```

- The general format of `scanf` is:

```
scanf(" [formatting] ", [arguments]);
```

- But the [formatting] should be minimal, and the [arguments] must be addresses. For example, to input an integer, save it in `num`, and print the input value:

```
int num;
scanf("%d", &num);
printf("%d", num);
```

Declaration and initialization

- The following code block declares an integer variable `num`:

```
int i;
```

- The variable `num` has no value until one is assigned:

```
int i;
i = 100;
```

- Alternatively, you can declare and initialize the variable in the same statement:

```
int i = 100;
```