Operating Systems

A (Soft) Introduction to C, Part I

Vassilis Markos, Mediterranean College

Week 06

Contents

Introduction to C++: `helloWorld.c`

Some Programming Humour, First...

MAN, I SUCK AT THIS GAME. CAN YOU GIVE ME A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

If you already understand this, you might be in the wrong room. Source: `https://xkcd.com/138/`.

helloWorld.c

```c
// source/helloWorld.c

#include <stdio.h> // Loads I/O functionality

int main() { // Main signature (it returns an integer)
    printf("Hello, world!\n"); // Prints to out stream
    return 0; // Mandatory, since main() returns an integer.
}
```

To execute: `gcc helloWorld.c -o helloWorld && ./helloWorld`

Notes on `helloWorld.c`

- `#include` is a Preprocessor command informing the preprocessor that it should “copy-paste” into this file the contents of the `stdio` library.
- Printing and IO, in general, is not built into C, as it is, e.g., in Python. So, we have to import `stdio.h` for that purpose.
- `return 0;` is a typical spell included in the end of every `main()` function.
    - We shall see in the near future that we can safely forget this in many cases, though.

Fancy String Characters in C++

Sequence	Meaning
`\a`	System bell (“beeps”)
`\b`	Backspace
`\f`	Page break (form feed)
`\n`	Line break (newline)
`\r`	Carriage return (returns the cursor to start of line)
`\t`	Tab
`\\`	Backslash
`\'`	Single quote
`\"`	Double quote
`\c`	Character represented by integer c

Frequently Used C Data Types

Type Name	Description	Size
`char`	Single text character, indicated with single quotes	1 byte
`int`	Signed or unsigned integer	4 bytes
`bool`[1]	Boolean	1 byte
`float`	Float number, 7 decimal digits accuracy	4 bytes
`double`	Double accuracy float, 15 decimal digits.	8 bytes

[1] Booleans are not a built-in feature of C; we need to include `stdbool.h`. Or just use `0` for `false` and `1` for `true`, alongside bitwise operators — your choice.

Frequently Used C++ Operators

Operator(s)	Description
`+`, `-`, `*`, `/`	Addition, subtraction, multiplication, division, priority determined as in maths (plus parentheses).
`%`	Modulus operator: Remainder of integer division, e.g., `13 % 4` evaluates to `1`.
`&&`, `||`, `!`	Logical `AND`, `OR`, `NOT`.
`==`, `!=`	`equals` and `not equals`.
`<`, `>`	`larger than` and `less than`.

Flow Control

Flow Control in C++

```c
// source/flowControl.c
#include <stdio.h>

int main() { // Main signature (it returns an integer)
    char knowsCpp; // Declaring a char variable
    printf("Do you use C? (y/n)\n");
    scanf(" %c", &knowsCpp); // Read single character from
   user.
    if (knowsCpp == 'y') { // Base case
        printf("Congrats! You already know C!\n");
    } else if (knowsCpp == 'n') { // If the above fails
        printf("Dont't worry, you can attend this course!\n");
    } else { // In case all of the above have failed
        printf("Please, enter 'y' or 'n'.\n");
    }
    return 0;
}
```

User Input In C

- To read user input, we used the following line of C code:

```c
    scanf(" %c", &knowsCpp); // Read single character from
     user.
```

- Breaking it down a bit:
  - `scanf()` is the `stdio.h` function that lets us read from the input stream.
  - The first argument is a *format specifier*, which explains to the compiler how to handle the provided string (array of characters).
  - The second argument is a pointer to the variable we want this piece of information to be stored at.
  - For the time being, you can think of the second argument as the target variable, prefixed by an ampersand (`&`).

Format Specifiers

- The general shape of a format specifier is: `%[*][width][length]specifier`.
- Some common format specifiers are:
  - `%d`: Signed integers, with any number of digits.
  - `%u`: Unsigned integers, with any number of digits.
  - `%f`: Floating point numbers, with any number of digits.
  - `%c`: Characters (no null termination here, see below).
  - `%s`: Null-terminated strings, including any character besides whitespace — so, the scanner stops at the first whitespace, introducing a `null` character there.

Format Specifiers

- In our case, we used a somewhat complex format specifier: `" %c"`.
- The `%c` part in the above is to read the next character entered by the user, as discussed above.
- Trying to remove the single blank space on the left, recompiling and executing the executable will result to actually prohibiting you from providing any input.
- That is because we use a *newline* character at the end of `printf()`, which is directly consumed by `scanf()` next.
- To prohibit that from happening, we add a blank space, indicating that `scanf()` should ignore any trailing whitespace.

Not relevant for the assessment of the module!

Format Specifier Fun

- A nice resource on format specifiers in C / C++ can be found here: https://cplusplus.com/reference/cstdio/scanf/.
- The format specification language is rich enough to be Turing complete[2], i.e., to allow for anything programmable to be programmed on it!
    - Available online here.
- An example of implementing Tic–Tac–Toe using just `printf()`: https://www.ioccc.org/2020/carlini/index.html

[2] Nicolas Carlini et al. “Control-flow bending: on the effectiveness of control-flow integrity”. In: *Proceedings of the 24th USENIX Conference on Security Symposium*. SEC’15. Washington, D.C.: USENIX Association, 2015, pp. 161–176. ISBN: 9781931971232.

`while` Loops in C

```c
// source/whileLoop.c
#include <stdio.h>

int main() {
    const int GUESS = 4; // Constant value (immutable)
    int x; // Declare variable
    printf("Guess what I'm thinking (int): ");
    scanf("%d", &x);
    while (x != GUESS) { // Repeat while condition holds
        printf("No, try again: ");
        scanf("%d", &x);
    }
    printf("Congratulations! You guessed right!\n");
    return 0;
}
```

for Loops in C++

```c
// source/forLoop.c
#include <stdio.h>

int main() {
    int x; // Declare variable
    int sum = 0; // Initialise variable
    for (int i = 0; i < 5; i++) {
    // for (index, terminating condition; step)
        printf("Please, enter an integer: ");
        scanf("%d", &x);
        sum = sum + x;
    }
    printf("Sum: %d.\n", x);
    return 0;
}
```

Useful Resources

Some resources you might find useful in your C++ journey:

- C Video Tutorial: https://www.youtube.com/watch?v=xND0t1pr3KY
- Learn-C Tutorial (interactive): https://www.learn-c.org/.
- Amazing C guide for anything in this course — and much more: https://beej.us/guide/bgc/.
    - You might also fancy the corresponding networks programming using C, by Beej: https://beej.us/guide/bgnet/.

Fun Time!

In–class Exercise #001

Write a C++ program that:

- asks the user for a positive integer number, n;
- checks if the number is even or odd, and;
- prints on screen `even` if the number is even, `odd` otherwise.

In–class Exercise #002

A PC manufacturer has the following retail pricing catalogue:

- For orders with at most 100 PCs, unit cost is 560$ each.
- For orders with at most 250 PCs, the first 100 are priced as above and the rest at 480$ each.
- For orders with at most 400 PCs, the first 250 are priced as above and the rest at 400$ each.
- For orders above 400 PCs, the first 400 are priced as above and the rest at 320$ each.

Write a C++ program that accepts the number of PCs ordered by some customer and computes and prints on screen to total cost of that order.

In–class Exercise #003

Write a C++ program that:

- asks the user for a positive integer number, n;
- checks if the number is prime or not, and;
- prints on screen `prime` if the number is prime, `composite` otherwise.

As a reminder, a positive integer, n, is said to be prime if the following conditions hold (both of them):

- $n > 1$.
- The only divisors of n are 1 and n.

So, 2, 7, 13 and 19 are some primes while 4, 15 and 21 are not.

In–class Exercise #004

Consider the C code shown right.

- Without compiling and executing it, what do you expect it to do?
- Compile and run that program. What did it print?
- Can you explain it?

```c
// source/exercise004.c
#include <stdio.h>

int main() {
    double x = 1.0;
    int i = 0;
    while (x > 0) {
        x = x / 2;
        i++;
    }
    printf("%f, %d\n", x, i);
    return 0;
}
```

In–class Exercise #005

Consider the C code shown right.

- Without compiling and executing it, what do you expect it to do?
- Compile and run that program. What did it print?
- Can you explain it?

```c
// source/exercise005
#include <stdio.h>
#include <stdbool.h>

int main() {
    double x = 1.0;
    while (x / 2 > 0) {
        x = x / 2;
    }
    double y = 1.4 * x;
    bool b = x == y;
    printf("%d\n", b);
    return 0;
}
```

Homework

Complete all exercises and problems in MIT's C++ course first assignment, found here:

`https://ocw.mit.edu/courses/6-096-introduction-to-c-january-iap-2011/resources/mit6_096iap11_assn01/`

For your convenience, you can also find the assignment file in this lecture's materials, at: `../homework/MIT6-096IAP11-assn01.pdf`.
Submit all your work in the online form below as a single `.zip` file:

`https://forms.gle/rSq3VSpcouRAVjqMA`

or via email at: `v.markos@mc-class.gr`.

Any Questions?

Do not forget to fill in the questionnaire shown right!

https://forms.gle/dKSrmE1VRVWqxBGZA