HANDS-ON C

Part I

Hello C

Hello World | Main-Function | Built-In Functionality | Translation Phase | Program Behavior

Hello World in C

- Most simple beginners program outputting „Hello World“
- Historic one (K&R C dialect), around 1978

material/Chapter 02/00 Hello World/main_KandR_C.c

```
main() {
    printf("Hello, World!\n“);
    return 0;
}
```

- Standard today

material/Chapter 02/00 Hello World/main_C11.c

```
#include <stdio.h>

int main(void) {
    printf("Hello, World!\n");
    return 0; // optional
}
```

- Compile source file and run the generated assembler out file

```
$ cc main_C11.c && ./a.out
Hello, World!
```

Inspect Hello World (1)

Program Entry for C Programs

```
return-type main( main parameter declaration )
{
    printf("Hello, World!\n");
    return return-value;
}
```

- Any executable written in C must contain the program entry point function `main`
  - normally function naming is up to programmer with exception of `main`
  - program execution starts at `main` function
  - `return-type` specifies the data type of the returned value (if any) when `return` statement is reached and program exits.
    - When left out, `int` type is implicitly be stated.
    - Specific for `main` : return type must be either
      - `void` if no value is returned, or
      - `int` a signed number indicating program state after termination

Inspect Hello World (2)

Program Entry for C Programs

```
return-type main( main parameter declaration )
{
    printf("Hello, World!\n");
    return return-value;
}
```

- `return return-value;` when control-flow reached this statement, a function returns the control-flow to the caller.
  - Specific for `main` : the program exists
  - `return` must not explicitly stated
    - but somehow required since C90 to avoid undefined behavior
    - alternatively, the program normally terminates by calling
      - `exit(int)` cleanup resources before exiting
      - `quick_exit(int)` don't cleanup resources completely before exiting
  - In case of non-empty return type (i.e., `int`), the following value can be returned
    - zero (macro `EXIT_SUCCESS`): successful exit
    - non-zero (macro `EXIT_FAILURE`): abnormal program termination

Inspect Hello World (3)

Program Entry for C Programs

```
return-type main( main parameter declaration )
{
    printf("Hello, World!\n");
    return return-value;
}
```

- `main parameter declaration` arguments given to a function by the caller.
  - Specific for `main`: program arguments for the executable
  - Two parameter declaration for main:
    - `void` indicates that program arguments are ignored
    - `int argc, char **argv` (alternatively `int argc, char *argv[]`)
      - argument count `argc` is number of program arguments
        - greater or equal 1
      - argument vector `argv` is list of strings where *i*-th string is *i*-th argument
        - first string is always absolute path to executable

Example: material/Chapter 02/00 Hello World/main_with_args.c

Inspect Hello World (4)

Program Entry for C Programs

```
return-type main( main parameter declaration )
{
   printf("Hello, World!\n");
   return return-value;
}
```

- `printf("Hello, World!\n");` is a call to a built-in („standard library“) function
  - it prints „Hello, World!“ followed by a new line to the output stream (stdout)
  - (Standard) library functions can be used „out of the box“
  - function definition (i.e., implementation) is in the (standard) library itself
  - function declaration (announcing that „there exists a specific function `printf` having that particular argument list“) is done in `stdio.h` (header) file
  - functions are made available in the program via (preprocessor) file inclusion

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