Week 2

- Selection
- Loops
- Something about preprocessor
- Important linux commands, getting help in linux (`man`)
- Something about pointers

`If` statement

The basic syntax of the `if` statement is:

```
if (condition)
{
    // code
}
```

- If `condition` is true, statements inside { and } are executed
- If `condition` is false, statements inside { and } are not executed

`if-else` statement

The if statement may have an optional else block:

```
if (condition) {
    // run code if condition is true
}
else {
    // run code if condition is false
}
```

Nested `if-else` statements

```
if (condition1) {                    // notice the parenthesis
   // statement(s)
}
else if (condition2) {               // like elif in Python
   // statement(s)
}
else if (condition3) {
   // statement(s)
}
. . .
else {                               // if all conditions above fail
   // statement(s)
}
```

File : `Example1.c`

Conditional `?:` operator

Syntax of a conditional operator:

```
variable = condition ? expr1 : expr2;
```

If `condition` is true, `variable` gets value `expr1`

If `condition` is false, `variable` gets value `expr2`

File: `Example2.c`

switch statement

`switch` allows a variable to be tested for equality against a list of values.

```
switch (variable) {

   case constant_1:
      statement(s);
      break;                // optional, but without behaves surprisingly!!

   . . .

   case constant_N:
      statement(s);
      break;

   default:
      statement(s);
}
```

File: `Example3.c`

While loop

```
while (condition){
  // the body of the loop
}
```

The **body** is executed as long as `condition` is true

```
 while (i <= 5) {
    printf("%d\n", i);
    i++;
  }
```

For-loop

```
for (initStatement; testExpression; updateStatement){
   // statements inside the body of loop
}
```

```
 for (i = 1; i <= 10; i++){
    printf("%d ", i);
  }
```

File: Example4.c

do - while

The `do - while` loop is similar to `while` with one important difference.

The body of `do - while` loop is executed at least once. Only then, the test `expression` is evaluated.

```
do {
   // the body of the loop
}
while (expression);
```

File: `Example5.c`

Other control statements

- `return` – ends the execution of a function, and returns control to the calling function
- `continue` – skips the current iteration of the loop and continues with the next iteration
- `break` – ends the loop/switch immediately when it is encountered
- `goto` (never use!)

Preprocessor

#include

Name	What it is for
stdio.h	Inputting and outputting, file handling
stdlib.h	General purpose functions, including dynamic memory management, random number generation, communication with the environment, integer arithmetics, searching, sorting and converting
string.h	defines several functions to manipulate C strings and arrays
time.h	to get and manipulate date and time information
math.h	common mathematical functions

Constants

Constants can be defined using preprocessor

```
#define MAX 100
```

- All "MAX" strings are replaced by 100 (like “search and replace” in an editor)

Another way to define constants: `const int size = 50;`

- using `const` we can control the scope. If it is placed inside any user-defined function, its effect is local
- If we have defined something using `#define`, that can be ***re-defined*** using `#undef` and then `#define`
- Try: `gcc -E preprocess.c > processed.c`

Common linux commands

cd	Change to directory.
clear	Clear a command line screen/window for a fresh start.
cp	Copy files and directories. `cd ..` goes to the parent directory
date	Display or set the system date and time.
grep	Search files or output for a particular pattern.
kill	Stop a process.
less	View the contents of a file one page at a time.
mv	Rename or move file(s) or directories.
rm	Remove (delete) file(s).
rmdir	Delete empty directories.

`man` command in linux

- Linux has a comprehensive documentation
- For example: `man gcc, man ls`
- Move: ***up-down arrows*** and jump by ***space bar***. Quit using ***q***
- `man` has different sections and part 3 is devoted to programming related issues
- If the exact name is unknown, use -k option

Example.

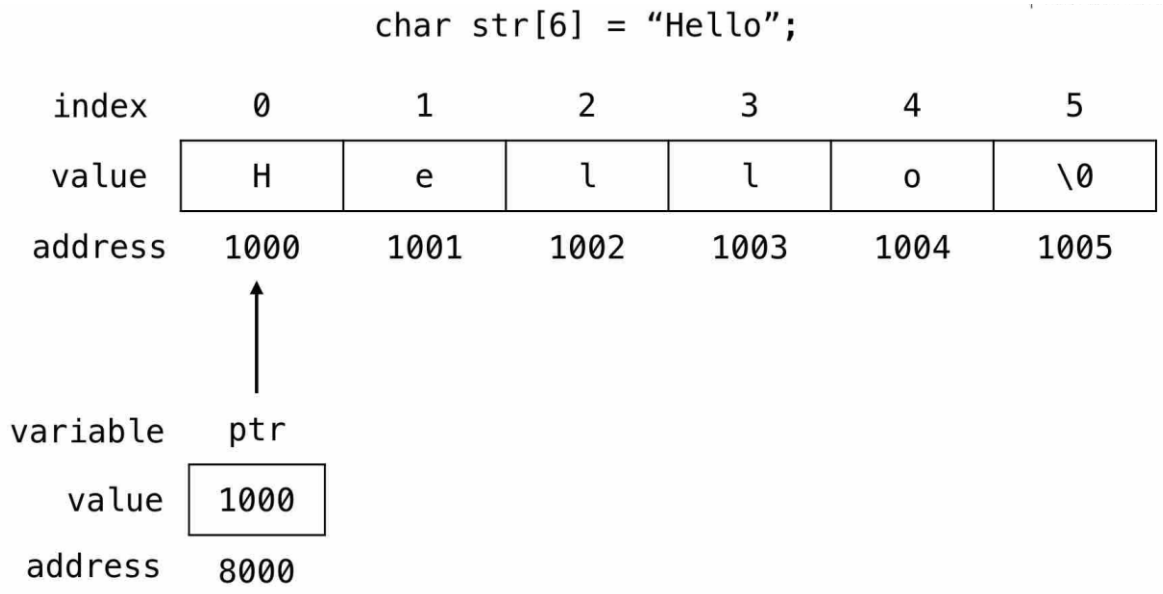
```
man -k print
man 3 printf  -- about outputting in C
man man
```

Addresses and pointers

File: `Example6.c` `//` If we use the & operator on a variable, it returns the memory address

---

The pointer can be declared using * (asterisk symbol): `char *ptr;`



Addresses and pointers

File: `Example7.c`

```
char *ptr;
```

- `*ptr` is a character,
- `ptr` is a pointer to a character

```
char str[] = "I love C language";
```

- ptr = str; // ptr points to the first character of the string // `str` is itself a pointer