Lecture 2_Arrays

Compiling

Compiler

1. *VS Code* of CS50 utilizes a compiler called `clang` or *c language*.

```
clang -o hello hello.c
```

```
Here, -o hello hello.c are provided as command - line arguments, which
represent outputting an executable file named hello from the source file
hello.c respectively.
```

2. When using some libraies, we need to provide that in command-line arguments.

```
clang -o hello hello.c -lcs50
```

```
Note that -l means library and cs50 is the library name
The clang automatically links to some libraries including stdio, so there is
no need to -lstdio
```

3. For convenience, we use `make`, it runs a command that executes clang to create an output file

4 steps in compiling

3. Preprocessing
    - Copy and paste header files into our file, in other words, code from the header files are copied into our program.
4. Compiling
    - Convert programming languega into assembly code
5. Assembing
    - Compiler converts assembly code into machine code
6. Linking
    - Code from your included libraries has also been converted into machine code and combined with your code. The final executable file is then outputted

Arrays

1. Each data type requires a certain amount of system resources:
   - `bool` 1 byte
   - `int` 4 bytes
   - `long` 8 bytes
   - `float` 4 bytes
   - `double` 8 bytes
   - `char` 1 byte
   - `string` ? bytes
2. Arrays are a sequence of values that are stored back to back in memory.
   `int scores[3]` is a way of telling the compiler to provide you three back-to-back places in memory of size `int` to store three `scores`
3. A string is an array of characters that ends with a NUL character

Command-Line Arguments

1. `Command-line arguments` are those arguments that are passed to your program at the command line.

```
int main(int argc, char * argv[])
```

- `int argc` is the number of command line arguments
- `char * argv[]` is an array of the characters passed as arguments at the command line.
- `char * argv[0]` is the name of the file, so `int argc` >= 1

Exit Status

1. When a program ends, a special exit code is provided to the computer.
   - When a program exits without error, a status code of `0` is provided to the computer
   - Often, when an error occurs that results in the program ending, a status of `1` is provided by the computer.
     Example:

```
// Returns explicit value from main

#include <cs50.h>
#include <stdio.h>

int main(int argc, string argv[])
{
    if (argc != 2)
    {
        printf("Missing command-line argument\n");
        return 1;
```

```
    }
    printf("hello, %s\n", argv[1]);
    return 0;
}
```

2. You can type `echo $?` in the terminal to see the exit status of the last run command.

Cryptography

1. *Encryption* is the act of hiding plain text from prying eyes.
2. *Decrypting* is the act of taking an encrypted piece of text and returning it to a human-readable form.
3. Cryptography is the art of ciphering and deciphering a message.

   `plaintext` and a `key` are provided to a `cipher`, resulting in ciphered text.