```
1. make hello
2. ./hello
```

Bunun ekvivalenti

```
1. clang hello.c
2. ./a.out
```

Make yazanda o avtomatik clang isledir. clang (c language). clang isletdikde, default adli olaraq a.out executable file'ini yaratmis olur.

clange argument olaraq -o (output), yazib ad qoyduqda, make ile olan kimi ./hello olur.

```
1. clang -o hello hello.c
2. ./hello
```

Cavabin float tipinde olmasini isteyirsense, hec olmasa bir dene float number involved olmalidir. Meselen 3 evezine 3.0 yazmaqla butun ifadenin cavabini float olmasini garanti etmek mumkundur. Ve ya Float kimi ifade etmek ucun bir variablei: (float) x

Array

arrayi declare etmek ucun birinci icindeki elementlerin hansi tipde oldugu, sonra arrayin adi, sonra [] icinde nece dene element olacagi yazilir.

```c
// Averages three numbers using an array, a constant, and a helper function

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

// Constant
const int N = 3;

// Prototype
float average(int length, int array[]);

int main(void)
{
    // Get scores
    int scores[N];
    for (int i = 0; i < N; i++)
    {
        scores[i] = get_int("Score: ");
    }

    // Print average
    printf("Average: %f\n", average(N, scores));
}
```

```c
float average(int length, int array[])
{
    // Calculate average
    int sum = 0;
    for (int i = 0; i < length; i++)
    {
        sum += array[i];
    }
    return sum / (float) length;
}
```

Bir arrayi funksiyaya argument kimi salanda, array[] seklinde yazilir, yeni icinde reqemsiz

Arrayin elementlerini tek-tek declare etmemek ucun :

```c
int hours[5] = {7, 9, 8, 7, 8};
```

Eger yuxaridaki kimi instantiation syntax yazilsa, o zaman [5] yazmaga arrayin olcusunu declare etmeye yoxdur.

Arraylari menimsetmek olmur birbirine, you can't deal an array like a variable. Kopyalamaq ucun asagidakini etmek lazimdir.

```c
int foo[5] = { 1, 2, 3, 4, 5 };
int bar[5];
for(int j = 0; j < 5; j++)
{
    bar[j] = foo[j];
}
```

Variable'da local olmayan kopyasinda deyisiklik olanda variable ozu deyismir, amma arrayda kopya goturulmur, funksiyaya direk olaraq array gonderilir. Ona gore de pramoy arrayin ustunde deyisiklik olur.

Strings

```c
int main(void)
{
    char c1 = 'H';
    char c2 = 'I';
    char c3 = '!';

    printf("%c%c%c\n", c1, c2, c3);
}
```

Bu sekilde yazanda, charlari print edir ve HI! Alinir.

```c
#include <stdio.h>

int main(void)
{
    char c1 = 'H';
```

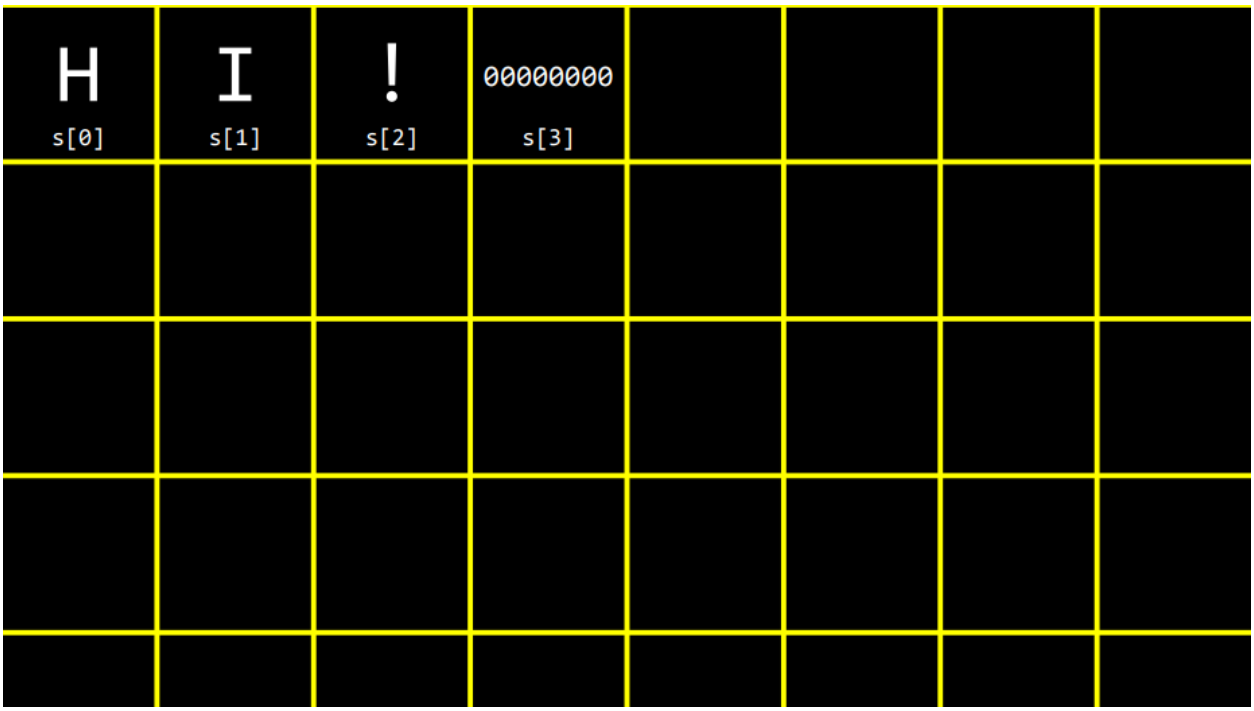
```
    char c2 = 'I';
    char c3 = '!';

    printf("%i %i %i\n", c1, c2, c3);
}
```

Burada ise charlarin ASCII-deki integer ekvivalentini print edir.

```
// Treats string as array

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

int main(void)
{
    string s = "HI!";
    printf("%c%c%c\n", s[0], s[1], s[2]); // HI!
}
```

Demeli string charlardan ibaret arraydir.

Memory'de, stringin qutardigi yerini belli etmek ucun her zaman len(s)+1 dene yer isledir register. Ve sonuncu yalniz 0lardan ibaret bayt olur. Bu 0 ededini ifade etmir. Special character olur ve \0 kimi ifade olunur. Sadece gosterir ki, string burada bitir. NUL-dur




```
// Prints string's ASCII codes, including NUL

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

```c
int main(void)
{
    string s = "HI!";
    printf("%i %i %i %i\n", s[0], s[1], s[2], s[3]); // 72 73 33 0
}
```

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

int main(void)
{
    string words[2]; //stringin array oldugunu nezere alsag, double array kimi olmus olur.

    words[0] = "HI!";
    words[1] = "BYE!";

    printf("%c%c%c\n", words[0][0], words[0][1], words[0][2]);
    printf("%c%c%c%c\n", words[1][0], words[1][1], words[1][2], words[1][3]);
}
```

Stringin uzunlugunu tapmaq ucun asagidaki kimi loop qurmaq olar, nul'dan istifade ederek.

```c
// Determines the length of a string

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

int main(void)
{
    // Prompt for user's name
    string name = get_string("Name: ");

    // Count number of characters up until '\0' (aka NUL)
    int n = 0;
    while (name[n] != '\0')
    {
        n++;
    }
    printf("%i\n", n);
}
```

Stringin lengthini tapmaq ucun olan funskiya – strlen() bunun ucun <string.h> librarysini elave etmek lazimdir

```c
#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Input: ")

    for (int i =0 ; i < strlen(s); i++)
    {
        // her dovrde tekrar tekrar strlen() funksiyasini icra edecek ona gore de
efficient deyil. 3.level
    }
}
```

```c
#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Input: ")
    int n = strlen(s)
    for (int i =0 ; i < n; i++)
    {
        // bu defe daha yaxsidir, cunki strlen() funksiyasi sadece bir defe icra
olunmus olacaq
        // amma yene de scope'unu variable'nin daha da daraltmaq mumkundur deye,
2.level dir
    }
}
```

```c
#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Input: ")

    for (int i =0, n = strlen(s); i < n; i++)
    {
```

```
        // bu defe lap daha yaxsidir, cunki scope daralmis oldu
        // ilk defe identitation edende icra olunu strlen() funksiyasi, also
        // yan yana coxlu variable declare etme temasini yada sal. ona gore
tekrar int yazilmayib.
        // 1.level efficiency
    }
}
```

in C, you can perform arithmetic operations between char and int types because characters are represented as integers in the ASCII table. Just remember that the result of such operations will also be an integer, so you may need to cast it back to a char if you want to use it as a character again. Numune below

```
// Uppercases a string

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

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (s[i] >= 'a' && s[i] <= 'z')
        {
            printf("%c", s[i] - 32);
          //printf("%c", s[i] - (‘a’ - ‘A’)); eyni seydi yuxaridakile
        }
        else
        {
            printf("%c", s[i]);
        }
    }
    printf("\n");
}
```

<ctype.h> - https://manual.cs50.io/#ctype.h

Yuxaridaki kodu, ctype librarysinden istifade ederek daha yigcam yazmaq mumkundur. toupper() funksiyasi her chari upper edir, ve upper olan charlari da handle edir. Yeni elave condition qurmaga ehtiyac olmur. Lowercaseleri upper edir ve qalan simvollar oldugu kimi qalir. Pramoy ustunde deyismir, value return edir.

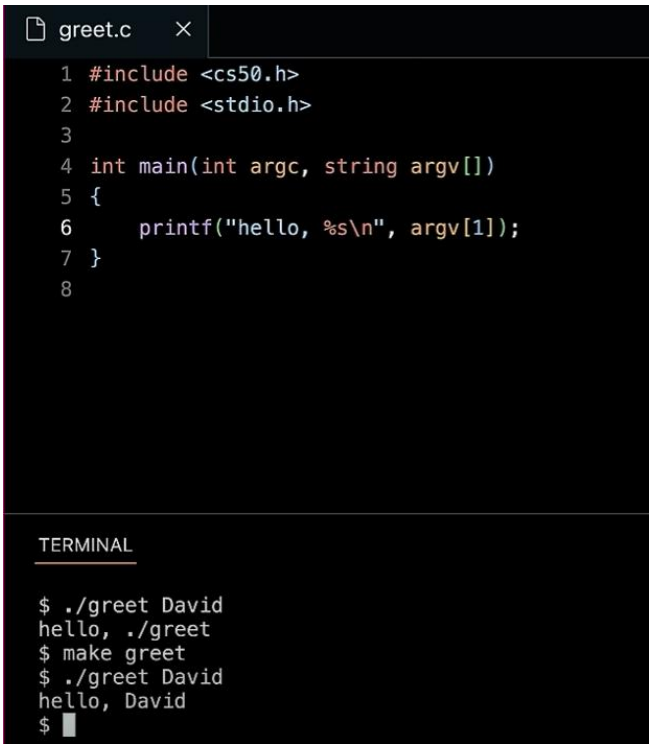
```c
// Uppercases string using ctype library

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

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c", toupper(s[i]));
    }
    printf("\n");
}
```

For comparing string we can't do it in like we do in Python and should rather use strcmp(arg1, arg2), is they are equal, it's gonna return 0. (string.h library)

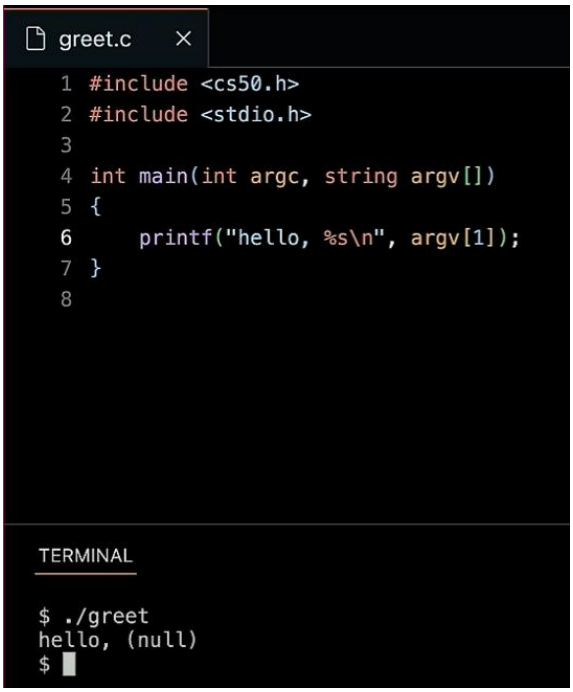
Command-line Arguments

```c
#include <stdio.h>
int main(int argc, string argv[])
{
 ...
}
```

Command-line'a access elemek ucun void evezine yuxaridaki kimi yazilir. argc – argument count, argv – argument vector (in this contest same as array) demekdir. argv'nin birinci elementinde file'in adi saxlanir. User'in prompt etdikleri 1ci indexde baslayir. Eger user nese prompt etmese (null) gorsenir. No value demekdir null, ve NUL'dan ferqlidir. Argv'nin icinde hersey string type'inda olur.



1ci defe, argv[0] print olundu, 2ci defe ise argv[1] hansiki userin prompt elediyi David gorsenir. 3cude ise nese prompt olmayib



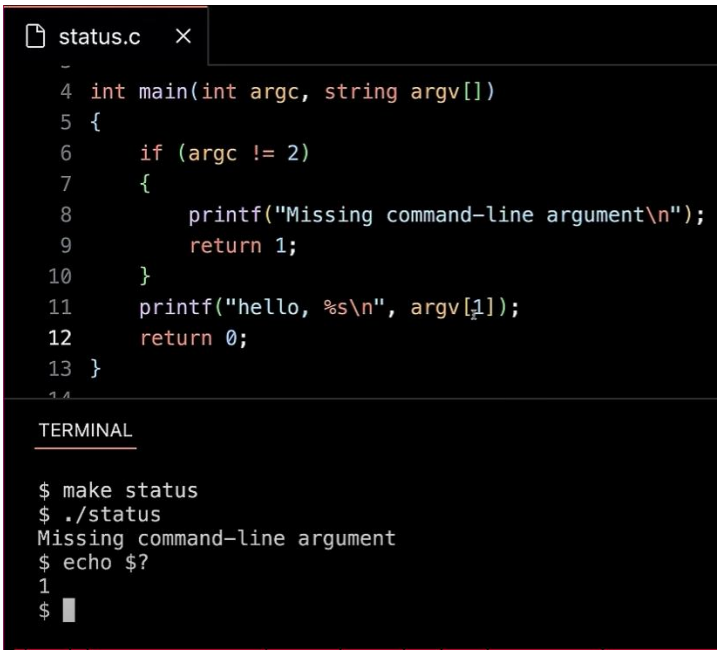
Bu kodda, user eger 1 soz yazsa, hemin yazilan soz+faylin adi birlikde len(argv)=2 olmus olur. Eger dha cox soz yazsa ve ya hec yazmasa default value print olmus olur.

```c
// Prints a command-line argument

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

int main(int argc, string argv[])
{
    if (argc == 2)
    {
        printf("hello, %s\n", argv[1]);
    }
```

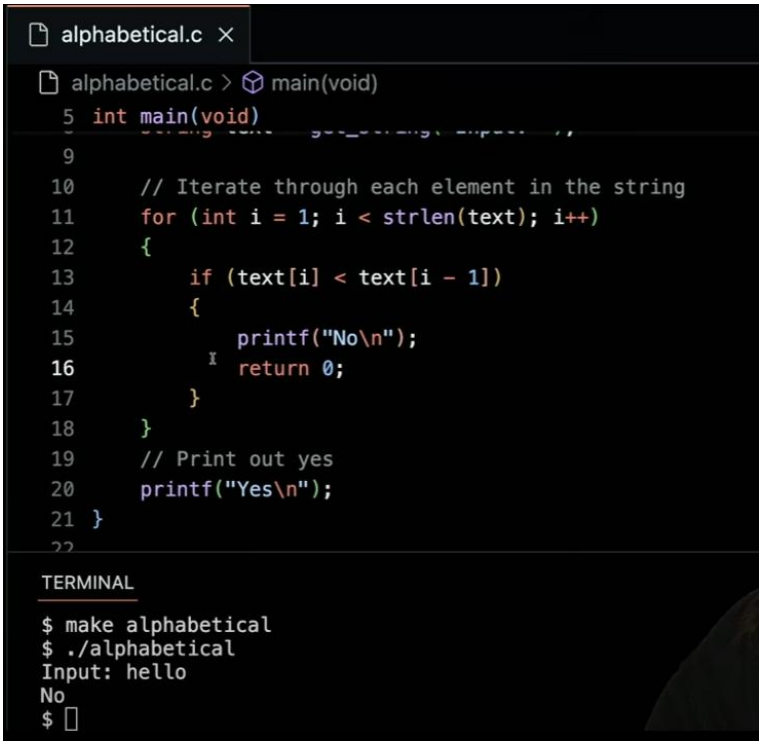
```
    else
    {
        printf("hello, world\n");
    }
}
```

Exit Status

When a program ends, a special exit code is provided to the computer. Numbers are used to standardize what can go wrong with the program. 0 means success by convention, but 1 means unsuccessful. Imagine exit status like error numbers. Like error 404.



int maindeki intin qaytardigi hemin bu exit statusdur. Direkt olaraq print elemir return value'ni. Onu gormek ucun echo $? Yazilir.



Exit statusdan elave ederek, break ve ya flagsiz sekilde dovr qurmaq mumkundur. Return value yazaraq. Burada elifba sirasi ile duzulmese string, return edib loopu qirir.

```c
#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

// Get user's input
int main(int argc, string argv[])
{
    // ask if user's input is one word, if not return an exit status
    if (argc != 2)
    {
        printf("Please provide a word.\n");
        return 1;
    }

    // declare the variables you gotta use multiple times
    string text = argv[1];
    int len = strlen(text);

    for (int i = 0; i < len; i++)
```

```
    {
        // ask from each char if it is letter in alphabet or not.
        // notice you don't ask it from the whole word like
unlike Python
        if (!isalpha(text[i]))
        {
            // raise second exit status if not letter
            printf("This is not a letter.\n");
            return 2;
        }
    }

    // Iterate through each element in the string
    for (int i = 1; i < len; i++)
    {
        // if the char is smaller than before, print NO
        if (text[i] < text[i - 1])
        {
            printf("No\n");
            return 0;
        }
    }
    // Print out yes
    printf("Yes\n");
}
```