CS240 Notes

Maninder (Kaurman) Kaur

June 19, 2025

Notes

Version 3

```
/* version 3 of z = x * y
   reads the numbers to be subtracted from keyboard
   using the standard I/O library function scanf()
   and outputs the result on the terminal
   using printf() */

#include <stdio.h>

int main()
{
    int x;
    int y, z;

    // read input
    scanf("%d %d", &x, &y);

    /* compute multiplication */
    z = x * y;

    // print result
    printf("%d * %d = %d\n", x, y, z);
}
```

`main()` calls `scanf()` to do something for it; the two inputs that should be read from the user should be stored into `int x` and `y`. This is done by putting every function from `MAIN MEMORY`, where they get their own working area. It is allocated for the function to use, allowing `main()` to call and use `scanf()`. Passing functions means to use them.

Alice and Bob are friends. She writes him two letters, placing them in mailbox 5 and 7 at the UPS office. Bob comes in later and opens 5 and 7 for

each letter. Alice and Bob represent the main function and the memory, while the letters are the functions.

Imagine memory as a bunch of slots that allow you to place data like bytes. Each slot allows 8 bits. The memory slots start at index 0 and go up to $2^n - 1$ slots. Integers take up 4 bytes.

How is this different than printf?

`main()` calls `printf()` to print on the terminal. It will print just the input of the variable. There is no need to store anything. `scanf` needs to know the address, while `printf` does not.

Segmentation Fault

A segmentation fault occurs when you try to access a data value that the OS does not give access to.

Version 4

```
/*      version 4 of z = x * y
        same as version 3 but supports
        real numbers */

#include <stdio.h>

int main()
{
    float x, y, z;

    // read input
    scanf("%f %f", &x, &y);

    // multiply
    z = x * y;

    // print result
    printf("result of %f times %f is %f\n", x, y, z);
}
```

Version 5

```
/*      version 5 of z = x * y
        same as version 4 but uses separate
```

```
        function multiply2() to perform multiplication */

#include <stdio.h>

float multiply2(float, float);

void main()
{
    float x, y, z;

    // read input
    scanf("%f %f", &x, &y);

    // compute
    z = multiply2(x, y);

    // print result
    printf("result of %f * %f is %.3f\n", x, y, z);
}

/*      function multiply2(a,b) takes two
        arguments of type float, multiplies a
        and b, and returns the result to
        the calling function */

float multiply2(float a, float b)
{
    float c;

    // multiply a with b
    // and store the result in local variable c
    c = a * b;

    // return value of c to calling function
    return c;
}
```

`printf()` works as follows: if there is a variable `x` and we assign it a value, to print it we would simply use `printf(%d, x)`. However, with `scanf`, we would use `scanf("%d", &x)`. We use & because we are not passing the value of `x`, but using the memory address itself to store the value.