

CS xyz3-001 Foundations of Programming and Data Structures

Instructor [Dr. Turgay Korkmaz](#)

Homework 03

Due date: check BB

!!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

Background (from our textbook (ROBERTS) ch-1 Problem 12.)

When you write a check, the dollar amount appears twice: once as a number and once as English text. For example, if you write a check for \$1729, you need to translate that number to the English text “**one thousand seven hundred twenty nine.**”

Your task in this problem is to write a program that reads in integers from the user and writes out the equivalent value in figures on the next line. Stop the program when the user enters any negative number. For example, the following is a sample run of this program:

Enter numbers in figures; use a negative value to stop.

Number: 0

zero

Number: 1

one

Number: 11

eleven

Number: 256

two hundred fifty six

Number: 1729

one thousand seven hundred twenty nine

Number: 2001

two thousand one

Number: 12345

twelve thousand three hundred forty five

Number: 13000

thirteen thousand

Number: -1

The key idea in this exercise is decomposition. The problem is not nearly as hard as it looks if you break it down into separate procedures that accomplish parts of the task. Many of these procedures will have a form that looks something like this:

```

void PrintOneDigit(int d)
{
    switch (d) {
        case 0: printf("zero"); break;
        case 1: printf("one"); break;
        case 2: printf("two"); break;
        case 3: printf("three"); break;
        case 4: printf("four"); break;
        case 5: printf("five"); break;
        case 6: printf("six"); break;
        case 7: printf("seven"); break;
        case 8: printf("eight"); break;
        case 9: printf("nine"); break;
        default: printf("Illegal call to PrintOneDigit");
    }
}

```

When writing your program, you should keep the following points in mind:

- Your program need work only with positive values up to 999,999. It should give the user some kind of error message if a number is outside of its range.
- You need to **get the number as integer** and **decompose** it into its digits! Don't try to get it as string or convert it to string. Specifically, get the number as follows and decompose it into digits:

```

int num;
int d6, d5, d4, d3, d2, d1;

printf("Enter Number : ");
scanf("%d", &num);
if (num > 999999) // give error ask another number
if(num < 0) // finish program

d1= num % 10;
d2 = (num % 100) / 10;
....

```

- Based on the values in d1, d2, d3, d4, d5, d6, all you have to do is display the corresponding text on the screen, which means that you need to call a function as in the description with digit and use **printf** to print hundred thousand etc after that digit.
- You should remain on the lookout for functions that you can reuse. For example, printing the number of thousands is pretty much the same as printing out the last three digits, and you should be able to use the same procedure more than once.

- Several special cases arise in this problem. For example, the number 11 must be treated differently than 21 or 31, because *eleven* doesn't fit the pattern established by *twenty one* and *thirty one*.
 - It is perfectly acceptable for all the letters in the output to be lowercase. The problem is much harder if you try to capitalize the first word, which you can try on your own time!
-

What to do and return: !!!! NO LATE HOMEWORK WILL BE ACCEPTED!!!!

1. Create a directory: `abc123-hw03`, using your own `abc123`. Do all your work under that directory.

2. Follow the problem-solving methodology to solve the problem(s). Then convert your solution(s) to a C program. You can name your program here as `hw03.c`

```
/*  
 * Don't forget to include comments about the  
 * problem, yourself and each major step in your  
 * program! so that we can understand your  
 * solution(s).  
 */
```

3. Compile and run your program. Copy/paste the results in an output file, which you can name as `hw03-out.txt`.

4. Zip the whole directory `abc123-hw03` as `abc123-hw03.zip`

5. Go to BB Learn (<http://learn.utsa.edu/>) , login using your `abc123`

6. Submit your `abc123-hw03.zip` for hw03 under Assignments

You must submit your work using Blackboard Learn and respect the following rules:

- 1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
- 2) Assignments must include all source code.
- 3) Assignments must include an `output.txt` file which demonstrates the final test output run by the student.
- 4) If your assignment does not run/compile, the `output.txt` file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.

[OPTIONAL IMPROVEMENTS]

Or you can also use array of strings to shorten your program like the following

```
char *ones[10]={ "zero", "one", "two", "three", "four", "five",  
"six", "seven", "eight", "nine"};  
  
void PrintOneDigit(int d)  
{  
    if (d>=0 && d<=9)  
        printf("%s", ones[d]);  
    else  
        printf("Illegal call to PrintOneDigit");  
}
```