Documentation of Chapter 2
Md. Mahfuj Hasan Shohug
BDCOM0019

1. Exercise 3-1:
   Our binary search makes two tests inside the loop, when one would suffice (at the price of more tests outside.) Write a version with only one test inside the loop and measure the difference in run-time.
   Source code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAXLEN 50000 // maximum length of array
#define ITERATIONS 500000  // iteration size
/*****************************************************************************************
** Functions     : main, int_array, book_binarysearch                                   **
**               : update_binarysearch, calculate_runtime                                **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters **
** Variables     : array[]        -- arry for put the number of list                     **
**               : search_value   -- searching value                                     **
**               : high_value     --  highest value of index                             **
**               : low_value      --  lowest value of index                              **
** Return        : = 0            -- Success                                             **
**               : < 0            -- Failed                                              **
** Note          : binary search with two version only one test inside the loop and     **
**                 measure the difference in run-time.                                  **
*****************************************************************************************/

/*function to making an arry for binary search*/
int int_array(int array[], int len)
{
    int i;
    for(i = 0; i>len; ++i)
    {
        array[i] = i;
    }
    return 0;
}

/*binary search given on the book*/
int book_binarysearch(int search_value, int array[], int len)
{
    int low_value, high_value, mid;
    low_value = 0;
    high_value = len - 1;
    while(low_value <= high_value)
    {
        mid = (low_value + high_value) / 2;
        if(search_value < array[mid])
        {
            high_value = mid - 1;
        }
        else if(search_value > array[mid])
        {
            low_value = mid + 1;
```

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Users\user\Desktop\test.exe
- Output Size: 154.564453125 KiB
- Compilation Time: 0.19s
```

D:\Repository\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-1.c - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Debug

(globals)

Project Classes Debug | Exercise 3-1.c | test.c | Exercise 3-2.c | Exercise 3-3.c

```c
        else if(search_value > array[mid])
        {
            low_value = mid + 1;
        }
        else
        {
            return mid;
        }
    }
    return -1;
}

/* update code for binary search */
int update_binarysearch(int search_value, int array[], int len)
{
    int low_value, high_value, mid;
    low_value = 0;
    high_value = len - 1;
    while (low_value <= high_value)
    {
        mid = (low_value + high_value) / 2;
        if (search_value < array[mid])
        {
            high_value = mid - 1;
        }
        else
        {
            low_value = mid + 1;
        }
    }
    if(search_value == array[low_value -1])
    {
        return low_value - 1;
    }
    return -1;
}

/* calculate binary search  run time */
int calculate_runtime(int test_binarysearch(int search_value, int array[], int len), int search_value, int array[], int len)
{
    static int num_test = 0;
    long clock_time = clock(); // Built in functions in the time.h library

    int i;
    for(i =0; i < ITERATIONS; ++i)
    {
        test_binarysearch(search_value, array, len);
    }
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Users\user\Desktop\test.exe
- Output Size: 154.564453125 KiB
- Compilation Time: 0.19s
```

```c
    return -1;
}

/* calculate binary search  run time */
int calculate_runtime(int test_binarysearch(int search_value, int array[], int len), int search_value, int array[], int len)
{
    static int num_test = 0;
    long clock_time = clock(); // Built in functions in the time.h library

    int i;
    for(i =0; i < ITERATIONS; ++i)
    {
        test_binarysearch(search_value, array, len);
    }
    clock_time = clock() - clock_time;
    printf("Run Time Test-%d: %lu Clocks (%.4f seconds)\n", num_test, clock_time, (double)clock_time / CLOCKS_PER_SEC);
    ++num_test;
    return 0;
}

/*main function*/
int main(int argc, char *argv[])
{
    int array[MAXLEN];
    int_array(array, MAXLEN); // call array making function

    int search_value = -1; //searching -1 for not found and complete the all iteration

    calculate_runtime(book_binarysearch, search_value, array, MAXLEN); // calculate runtime for on book program
    calculate_runtime(update_binarysearch, search_value, array, MAXLEN); // calculate runtime for modify program
}
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Users\user\Desktop\test.exe
- Output Size: 154.564453125 KiB
- Compilation Time: 0.19s
```

Here on this code I am calculating the runtime and compare to book and my code. Here I am analyzing some random search input and show the output.
When search_value = -1

```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-1.exe
Run Time Test-0: 26 Clocks (0.0260 seconds)
Run Time Test-1: 27 Clocks (0.0270 seconds)

--------------------------------
Process exited after 0.0853 seconds with return value 0
Press any key to continue . . .
```

For book: 0.0260 sec
And modify: 0.0260 sec

When search_value: 25000

```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-1.exe
Run Time Test-0: 32 Clocks (0.0320 seconds)
Run Time Test-1: 28 Clocks (0.0280 seconds)

--------------------------------
Process exited after 0.09019 seconds with return value 0
Press any key to continue . . .
```

For book its 0.0320 sec
And for modify: 0.280 sec.
So that the modify code will be most optimize.

2. Exercise 3-2:
Write a function escape(s,t) that converts characters like newline and tab into visible escape sequences like \n and \t as it copies the string t to s. Use a switch. Write a function for the other direction as well, converting escape sequences into the real characters.
Source Code:

D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-2.c - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Debug

(globals)

Project Classes Debug | Exercise 3-1.c | test.c | Exercise 3-2.c | Exercise 3-3.c

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

#define MAXLEN 1000 //maximum array length
/*****************************************************************************************
** Functions     : main, escape, unescape, get_input                                    **
**                                                                                       **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**               : 2. argv        -- The pointer to the input string array of parameters  **
** Variables     : s[]            -- arry for char value                                  **
**               : t[]            -- arry for the char value                              **
**                                                                                       **
** Return        : = 0            -- Success                                              **
**               : < 0            -- Failed                                               **
** Note          :  converting escape sequences into the real characters                 **
*****************************************************************************************/

/*visible escape sequences*/
void escape(char s[], char t[])
{
  int i, j;
  for (i = j = 0; t[i] != '\0'; ++i, ++j)
  {
    switch (t[i])
    {
    case '\a':
      s[j++] = '\\';
      s[j] = 'a';
      break;

    case '\b':
      s[j++] = '\\';
      s[j] = 'b';
      break;

    case '\f':
      s[j++] = '\\';
      s[j] = 'f';
      break;

    case '\n':
      s[j++] = '\\';
      s[j] = 'n';
      break;

```

D:\Repository\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-2.c - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Debug

(globals)

Project Classes Debug | Exercise 3-1.c | test.c | Exercise 3-2.c | Exercise 3-3.c

```c
        s[j] = 'n';
        break;

      case '\r':
        s[j++] = '\\';
        s[j] = 'r';
        break;

      case '\t':
        s[j++] = '\\';
        s[j] = 't';
        break;

      case '\v':
        s[j++] = '\\';
        s[j] = 'v';
        break;

      case '\\':
        s[j++] = '\\';
        s[j] = '\\';
        break;

      case '\?':
        s[j++] = '\\';
        s[j] = '?';
        break;

      case '\'':
        s[j++] = '\\';
        s[j] = '\'';
        break;

      case '\"':
        s[j++] = '\\';
        s[j] = '"';
        break;

      default:
        s[j] = t[i];
        break;
      }
    }

    if (t[i] == '\0')
    {
      s[j] = t[i];
    }
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Repository\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-1.exe
- Output Size: 156.5390625 KiB
- Compilation Time: 0.19s
```

D:\Repository\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-2.c - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Debug

(globals)

Project Classes Debug | Exercise 3-1.c | test.c | Exercise 3-2.c | Exercise 3-3.c

```c
        s[j] = '\\';
        break;

      case '?':
        s[j] = '\?';
        break;

      case '\'':
        s[j] = '\'';
        break;

      case '"':
        s[j] = '\"';
        break;

      default:
        s[j++] = '\\';
        s[j] = t[i];
        break;
      }
      break;

    default:
      s[j] = t[i];
      break;
    }
  }

  if (t[i] == '\0')
  {
    s[j] = t[i];
  }
}

/*input char*/
int get_input(char line[], unsigned int limit)
{
  int i, c;
  for (i = 0; i < limit - 1 && (c = getchar()) != EOF && c != '\n'; ++i)
  {
    line[i] = c;
  }

  if (c == '\n')
  {
    line[i++] = c;
  }
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Repository\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-1.exe
- Output Size: 156.5390625 KiB
- Compilation Time: 0.19s
```

```c
/*input char*/
int get_input(char line[], unsigned int limit)
{
  int i, c;
  for (i = 0; i < limit - 1 && (c = getchar()) != EOF && c != '\n'; ++i)
  {
    line[i] = c;
  }

  if (c == '\n')
  {
    line[i++] = c;
  }

  line[i] = '\0'; // for null value

  return i;
}

/*main function*/
int main(int argc, char *argv[])
{
    char t[MAXLEN], s[MAXLEN];;

    get_input(t, MAXLEN);

    escape(s, t);
    printf("%s\n", s); // print  escape sequences into the real characters

    unescape(s, t);
    printf("%s", s); // remove escape sequences real characters and print

    return 0;
}
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

Abort Compilation

Shorten compiler paths

```
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-1
- Output Size: 156.5390625 KiB
- Compilation Time: 0.19s
```

The user's original input string is displayed so that you can see what they typed in. The function that mirrors the same string is run by the program, revealing any escape characters that were utilized. The program then displays the initial input string as the result once more, making it clear what happened. Using a switch-case statement, the escape characters are found. Every case in the switch statement determines whether the character matches a certain escape character and then does the necessary action.

The program's overall goal is to display the input string, draw attention to any escape characters utilized, then print the original input string in order to give users a complete understanding of the processes carried out.

For input: “Hello word”

D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-2.exe

```
Hello           word
Hello\t\tword\n
Hello           word

--------------------------------
Process exited after 7.766 seconds with return value 0
Press any key to continue . . .
```

For “Mahfuj Hasan ”
Output:

```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-2.exe
Mahfuj hasan
Mahfuj hasan \t\t\t\n
Mahfuj hasan

--------------------------------
Process exited after 7.01 seconds with return value 0
Press any key to continue . . .
```

3. Exercise 3-3:
Write a function expand(s1,s2) that expands shorthand notations like a-z in the string s1 into the equivalent complete list abc...xyz in s2. Allow for letters of either case and digits, and be prepared to handle cases like a-b-c and a-z0-9 and -a-z. Arrange that a leading or trailing - is taken literally.
Code File:

```c
#include <stdio.h>
#include <ctype.h>
/*****************************************************************************************
** Functions     : main, expand,                                                          **
**                                                                                        **
** Inputs        : 1. argc        -- The number of parameters provided to the main function**
**                : 2. argv        -- The pointer to the input string array of parameters  **
**                                                                                        **
** Return        : = 0            -- Success                                              **
**                : < 0            -- Failed                                               **
** Note          :  converting escape sequences into the real characters                  **
*****************************************************************************************/
/* declire expand function*/
void expand(const char s1[], char s2[]);

/*main func*/
int main(void)
{
    const char s1[] = "a-z 1-5 A-D";
    char s2[1000];

    expand(s1, s2);
    printf("Expanded string: %s\n", s2);

    return 0;
}

void expand(const char s1[], char s2[])
{
    int i, j, k;

    i = j = 0;

    while (s1[i] != '\0') {
        if (s1[i] == '-' && i > 0 && s1[i + 1] != '\0') {
            char start = s1[i - 1];
            char end = s1[i + 1];

            if ((islower(start) && islower(end)) || (isupper(start) && isupper(end)) || (isdigit(start) && isdigit
                for (k = start + 1; k <= end; k++) {
                    s2[j++] = k;
                }
            } else {
                s2[j++] = s1[i];
            }
        } else {
            s2[j++] = s1[i];
        }

        i++;
    }

    s2[j] = '\0';
}
```

Output:

```
D:\Reposetory\Training\MdMahfujHasanShohug\C&DS\Day_4\Exercise 3-3.exe
Expanded string: abcdefghijklmnopqrstuvwxyzz 123455 ABCDD

--------------------------------
Process exited after 0.03099 seconds with return value 0
Press any key to continue . . .
```

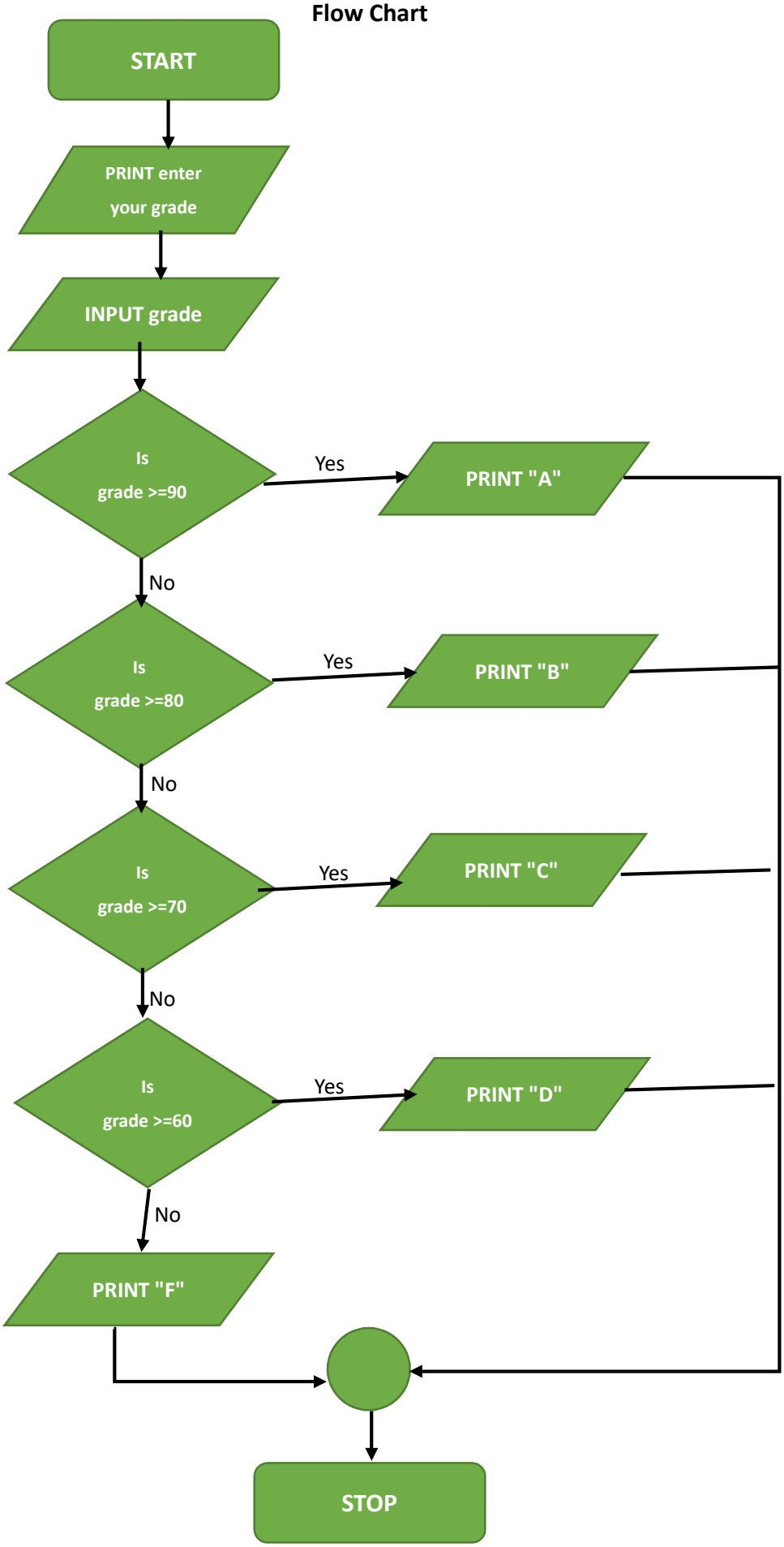
The software provides an example of how to calculate a string's length. The program uses a string of lowercase letters from "a" to "z" as an example. The ASCII values of the characters are used by the algorithm to automatically carry out the calculation.
This is accomplished via the program's use of an iterative loop that goes from the ASCII value of "a" to the ASCII value of "z." The string's length is increased with each repetition. This procedure makes sure that the program determines the string's length precisely.
Similar to lowercase letters, the program uses the matching ASCII values to determine the length of the string for uppercase letters.
The show demonstrates how to measure the length of a string by

4. Exercise chapter-2 question:
   Represent the following Pseudocode as a flowchart:

```
If student's grade is greater than or equal to 90
    Print "A"
Else If student's grade is greater than or equal to 80
    Print "B"
Else If student's grade is greater than or equal to 70
    Print "C"
Else If student's grade is greater than or equal to 60
    Print "D"
else
    Print "F"
```

Description: Different types of shapes on the flow chart here allude to different types of grades where the number matters on the grade. several meanings, some referring to input and output, others to check conditions, etc. The condition determines if the supplied statement is true or false; if true, the statement is executed; if false, the following condition is checked and executed.

Flow Chart


START
PRINT enter your grade
INPUT grade
Is grade >=90
Yes
PRINT "A"
No
Is grade >=80
Yes
PRINT "B"
No
Is grade >=70
Yes
PRINT "C"
No
Is grade >=60
Yes
PRINT "D"
No
PRINT "F"
STOP