Learning summary:
Study: 5 Hours
Exercises: 2.5 Hours

Documentation of Day_18

Exercise 5-3:

Write a pointer version of the function strcat that we showed in Chapter 2: strcat(s,t) copies the string t to the end of s.

Source Code:

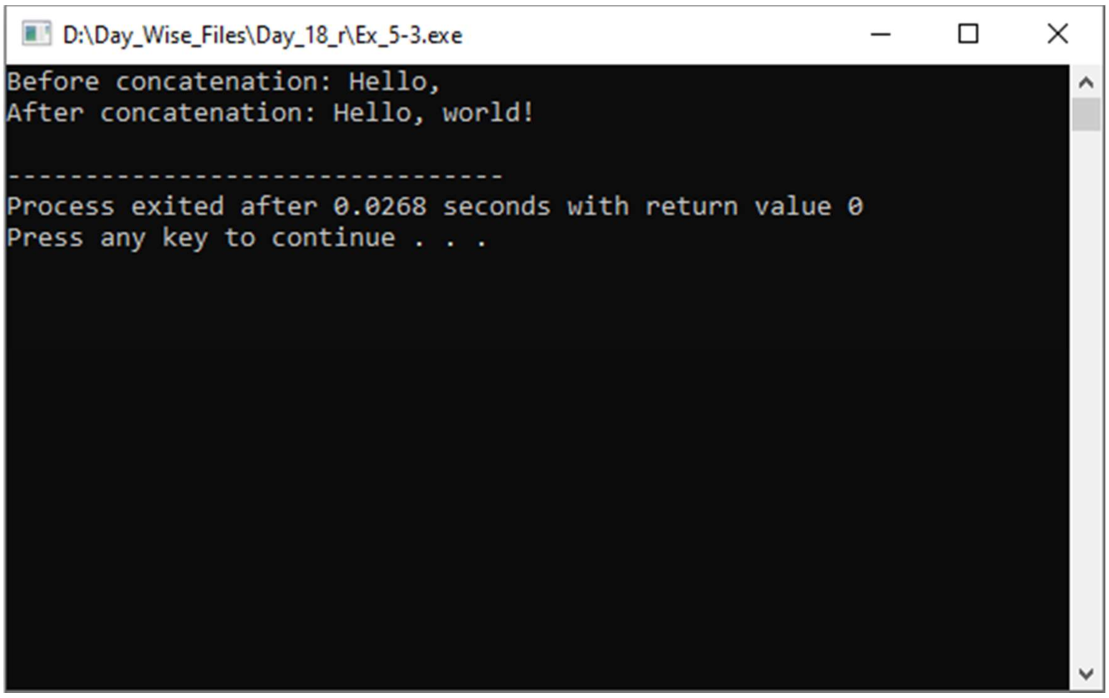
```
#include <stdio.h>
void my_strcat(char *s, const char *t) {
        while (*s != '\0') {
                s++;
        }

        while ((*s = *t) != '\0') {
                s++;
                t++;
        }
}

int main() {
   char s[100] = "Hello, ";
   char t[] = "world!";

   printf("Before concatenation: %s\n", s);
   my_strcat(s, t);
   printf("After concatenation: %s\n", s);

   return 0;
}
```

In this code I have defined a function strcat that concatenates two strings. It takes two arguments: s, a pointer to a character array (string) where the result will be stored, and t, a pointer to a constant character array (string) that will be appended to s.

The function first moves the pointer s to the end of the string s by incrementing it until it reaches the null character '\0' indicating the end of the string.

Then it enters a loop where it assigns each character of t to the current position in s using the assignment operator *s = *t. It increments both s and t after each assignment. The loop continues until it encounters the null character in t, and then it stops, effectively appending t to the end of s.

Output:



```
Before concatenation: Hello,
After concatenation: Hello, world!

--------------------------------
Process exited after 0.0268 seconds with return value 0
Press any key to continue . . .
```

Exercise 5-5:

Write versions of the library functions strncpy, strncat, and strncmp, which operate on at most the first n characters of their argument strings. For example, strncpy(s,t,n) copies at most n characters of t to s.

Source Code:

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

char* my_strncpy(char *dest, const char *src, size_t n)
{
  char *dest_start = dest;
  size_t i;

  for (i = 0; i < n && *src != '\0'; i++) {
    *dest++ = *src++;
  }

  for (; i < n; i++) {
    *dest++ = '\0';
  }

  return dest_start;
```

```
}

char* my_strncat(char *dest, const char *src, size_t n)
{
    char *dest_start = dest;

    while (*dest != '\0') {
        dest++;
    }

    size_t i;
    for (i = 0; i < n && *src != '\0'; i++) {
        *dest++ = *src++;
    }

    *dest = '\0';

    return dest_start;
}

int my_strncmp(const char *s1, const char *s2, size_t n)
{
    size_t i;
    for (i = 0; i < n; i++) {
        if (*s1 != *s2) {
            return (*s1 - *s2);
        }
        else if (*s1 == '\0') {
            return 0;
        }

        s1++;
        s2++;
    }

    return 0;
}

int main() {
    char source[] = "Hello, world!";
    char destination[10];

    my_strncpy(destination, source, 5);
    destination[sizeof(destination) - 1] = '\0';
    printf("Original String: %s\n", source);
            printf("The function 'strncpy' Invoked!\n");
            printf("Result: %s\n\n", destination);

            char source1[] = "Hello, ";
    char destination1[] = "world!";
    printf("String1 = %s String2 = %s\n", source1, destination1);
            my_strncat(destination1, source1, strlen(source1));
            printf("The function 'strncat' Invoked!\n");
            printf("Result: %s\n\n", destination1);

            char str1[] = "Hello";
    char str2[] = "Hell";
    printf("str1 = %s, str2 = %s\n", str1, str2);
    printf("The function 'strncat' Invoked!\n");
            int result = my_strncmp(str1, str2, 5);
    if (result < 0) {
       printf("str1 is less than str2\n");
    }
    else if (result > 0) {
       printf("str1 is greater than str2\n");
    }
    else {
       printf("str1 is equal to str2\n");
    }

    return 0;
}
```

1. **my_strncpy:**

- This function takes three arguments: dest (destination string), src (source string), and n (maximum number of characters to copy).
- It copies at most n characters from src to dest.
- It also ensures that dest is null-terminated by appending null characters if necessary.
- It returns a pointer to the modified dest string.

2. **my_strncat:**

- This function takes three arguments: dest (destination string), src (source string), and n (maximum number of characters to concatenate).
- It appends at most n characters from src to the end of dest.
- It ensures that dest remains null-terminated after the concatenation.
- It returns a pointer to the modified dest string.

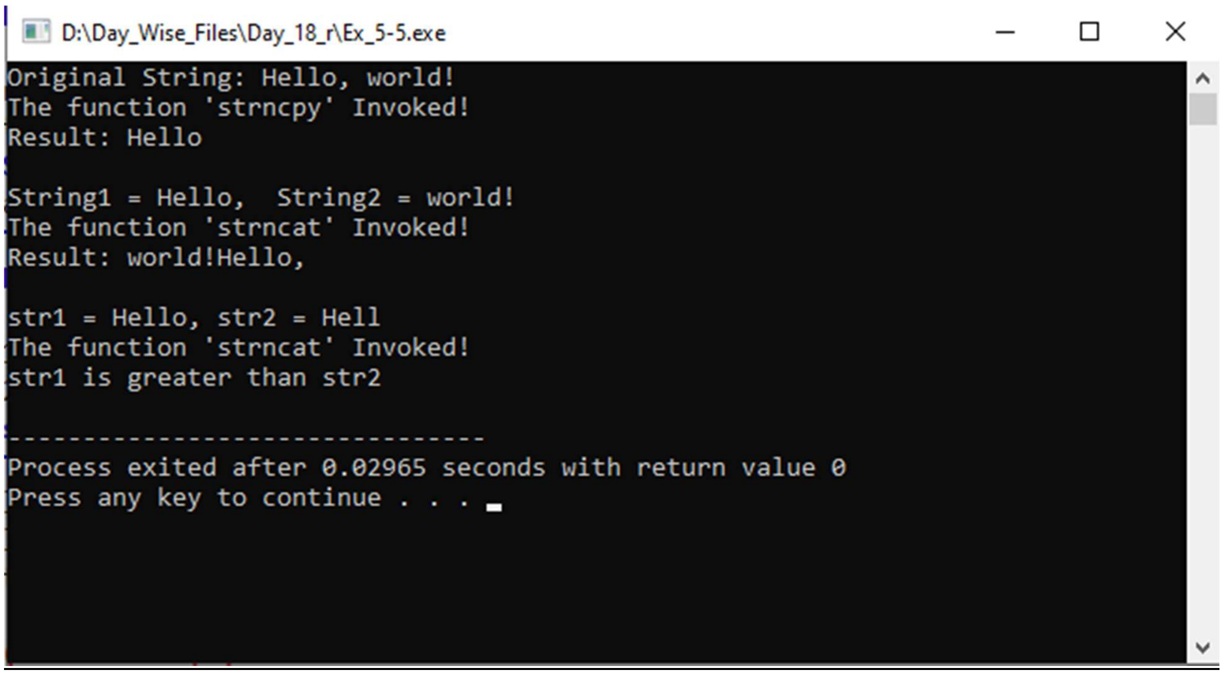
3. **my_strncmp:**

- This function takes three arguments: s1 (first string to compare), s2 (second string to compare), and n (maximum number of characters to compare).
- It compares at most n characters between s1 and s2.
- It returns a negative value if s1 is lexicographically less than s2, a positive value if s1 is lexicographically greater than s2, or 0 if they are equal.

4. main function:

- The main function demonstrates the usage of the above functions with example scenarios.
- It creates a source string (source) and a destination string (destination), and calls my_strncpy to copy a maximum of 5 characters from source to destination.
- It then prints the original string (source) and the modified string (destination).
- Next, it showcases the usage of my_strncat by appending source1 to the end of destination1.
- It prints the result of the concatenation.
- Finally, it demonstrates the usage of my_strncmp by comparing str1 and str2 for the first 5 characters, and prints the result of the comparison.

Input and Output:



D:\Day_Wise_Files\Day_18_r\Ex_5-5.exe

```
Original String: Hello, world!
The function 'strncpy' Invoked!
Result: Hello

String1 = Hello,  String2 = world!
The function 'strncat' Invoked!
Result: world!Hello,

str1 = Hello, str2 = Hell
The function 'strncat' Invoked!
str1 is greater than str2

--------------------------------
Process exited after 0.02965 seconds with return value 0
Press any key to continue . . .
```

Exercise 5-6
Rewrite appropriate programs from earlier chapters and exercises with pointers instead of array indexing. Good possibilities include getline (Chapters 1 and 4), atoi, itoa, and their variants (Chapters 2, 3, and 4), reverse (Chapter 3), and strindex and getop (Chapter 4).
Source Code:
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#define NUMBER '0'
#define MAX_INPUT_SIZE 100

int my_getline(char *s, int lim) {
  int c;
  char *s_start = s;

  while (--lim > 0 && (c = getchar()) != EOF && c != '\n') {
    *s++ = c;
  }

  if (c == '\n') {
    *s++ = c;
  }

  *s = '\0';

  return s - s_start;
}

int my_atoi(const char *s)
{
  int n = 0;
  int sign = 1;

  while (isspace(*s))
    s++;

  if (*s == '-' || *s == '+') {
    if (*s == '-')
      sign = -1;
    s++;
  }
```

```
    while (isdigit(*s)) {
        n = n * 10 + (*s - '0');
        s++;
    }

    return sign * n;
}

void reverse(char *s)
{
    char *start = s;
    char *end = s + strlen(s) - 1;

    while (start < end) {
        char temp = *start;
        *start = *end;
        *end = temp;

        start++;
        end--;
    }
}

void my_itoa(int n, char *s)
{
    int sign = 1;

    if (n < 0) {
        sign = -1;
        n = -n;
    }

    char *ptr = s;

    do {
        *ptr++ = n % 10 + '0';
        n /= 10;
    } while (n > 0);

    if (sign == -1) {
        *ptr++ = '-';
    }

    *ptr = '\0';
    reverse(s);
}

int my_strindex(char *s, char *t)
{
    char *s_ptr, *t_ptr, *start;

    for (s_ptr = s; *s_ptr != '\0'; s_ptr++) {
        start = s_ptr;
        t_ptr = t;

        while (*t_ptr != '\0' && *s_ptr == *t_ptr) {
            s_ptr++;
            t_ptr++;
        }

        if (*t_ptr == '\0')
            return start - s;

        s_ptr = start;
    }

    return -1;
}

int my_getop(char *s) {
    int i, c;

    while ((*s = c = getchar()) == ' ' || c == '\t')
        ;

    *(s + 1) = '\0';

    if (!isdigit(c) && c != '.')
        return c; /* not a number */

    if (isdigit(c)) {
        while (isdigit(*(++s) = c = getchar()))
            ;
    }

    if (c == '.') {
        while (isdigit(*(++s) = c = getchar()))
        ;
    }

    *s = '\0';

    if (c != EOF)
        ungetc(c, stdin);

    return NUMBER;
}

int main() {
    char line[100];
    int length;

    printf("Enter a line of text: ");
    length = my_getline(line, sizeof(line));
            printf("The function 'getline' Invoked!\n");
    printf("Line: %s", line);
    printf("Length: %d\n\n", length);

    const char *str = "   -12345";
    printf("Input for 'atoi' function: %s\n", str);
    int result = my_atoi(str);
    printf("The function 'atoi' Invoked!\n");
    printf("Result: %d\n\n", result);

    int num = -12345;
    char str1[20];
    printf("Input for 'itoa' function: %d\n", num);
    my_itoa(num, str1);
            printf("The function 'itoa' Invoked!\n");
    printf("Result: %s\n\n", str1);

    char s[] = "Hello, world!";
    char t[] = "world";
    printf("Source string: %s\n", s);
    printf("Substring: %s\n", t);
    printf("The function 'strindex' Invoked!\n");
            int index = my_strindex(s, t);
    if (index != -1) {
        printf("Substring found at index: %d\n", index);
    }
            else {
```

```
    printf("Substring not found\n");
  }

  char input[100];
          printf("The function 'getop' Invoked!\n");
  printf("Enter an expression: ");
  while (my_getop(input) != EOF) {
    printf("Operand or operator: ");
                    printf("%s", input);
    printf("\n");
  }
          return 0;
}
```
my_getline:

- This function takes two arguments: s (a character array to store the input line) and lim (the maximum size of the array).
- It reads characters from the standard input until it encounters EOF, a newline character, or the array size limit.
- It stores the input characters in the array s.
- It returns the length of the input line.

my_atoi:

- This function takes a string s as input and converts it to an integer.
- It skips leading whitespace characters and handles an optional sign.
- It iterates through the remaining characters, converting them to an integer.
- It returns the converted integer.

reverse:

- This function takes a string s as input and reverses its characters.
- It uses two pointers (start and end) to swap characters from opposite ends of the string until they meet in the middle.

my_itoa:

- This function takes an integer n and a character array s as input and converts the integer to a string.
- It handles negative numbers by storing the sign separately and converting the absolute value.
- It iteratively converts each digit of the integer to a character and stores it in the array s.
- It adds a '-' character if the number was negative.
- It null-terminates the resulting string.

my_strindex:

- This function takes two strings, s and t, as input and searches for the first occurrence of t within s.
- It iterates through s, comparing characters with t character by character.
- If a match is found, it returns the index of the start of the matching substring in s.
- If no match is found, it returns -1.

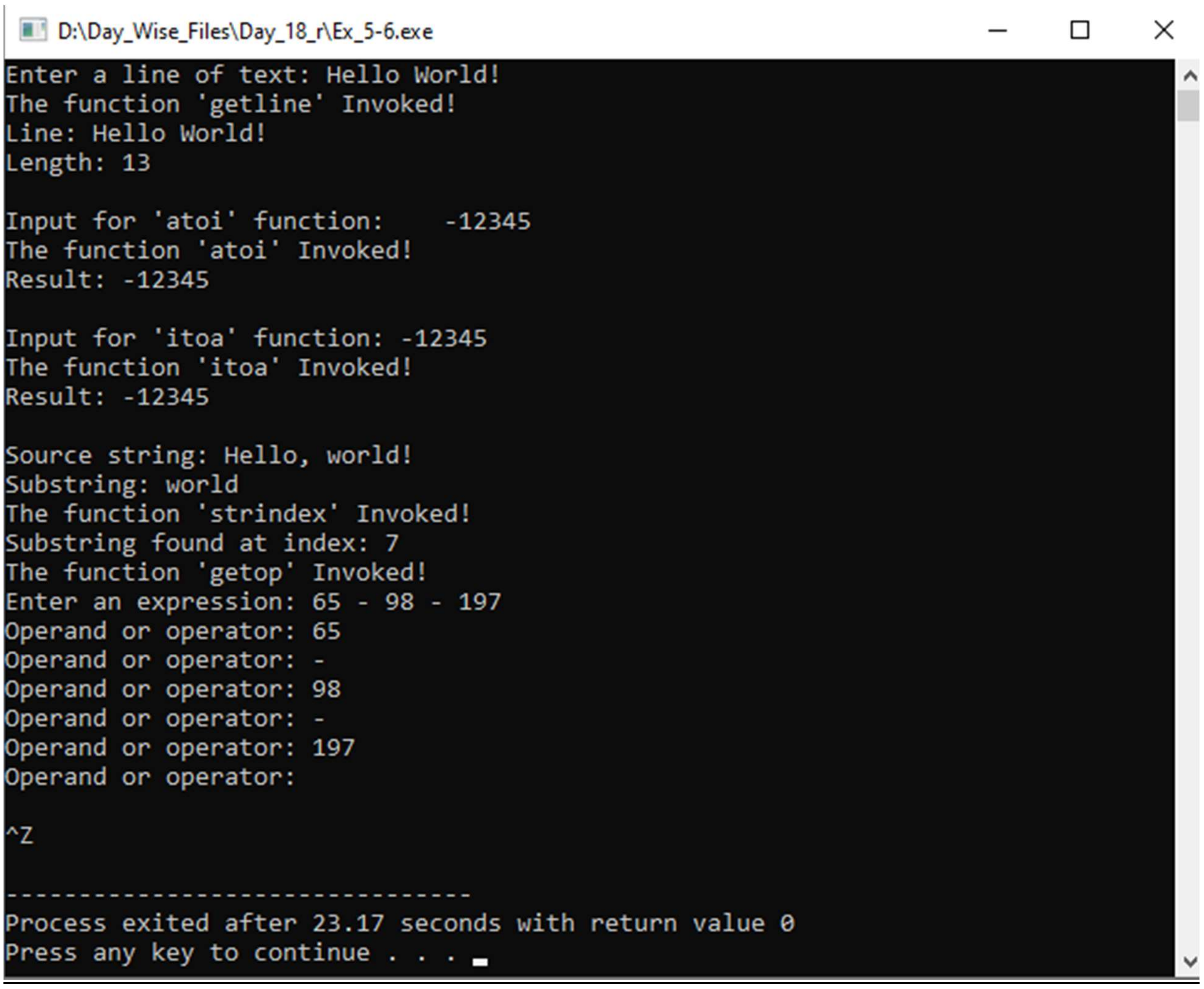
my_getop:

- This function reads a sequence of characters from the standard input and categorizes them as either numbers or operators.
- It skips leading whitespace characters.
- If the first character encountered is a digit or a '.', it reads subsequent characters until it no longer forms a valid number.
- It returns 'NUMBER' if a number is encountered, or the character itself if it is an operator.

main function:

- The main function demonstrates the usage of the above functions with example scenarios.
- It calls my_getline to read a line of text from the user and prints the line and its length.
- It showcases the usage of my_atoi by converting a string to an integer and printing the result.
- It demonstrates my_itoa by converting an integer to a string and printing the result.
- It showcases the usage of my_strindex by searching for a substring within a source string and printing the result.
- Finally, it demonstrates the usage of my_getop by reading and categorizing input expressions until EOF is encountered

<u>Input and Output:</u>



```
Enter a line of text: Hello World!
The function 'getline' Invoked!
Line: Hello World!
Length: 13

Input for 'atoi' function:    -12345
The function 'atoi' Invoked!
Result: -12345

Input for 'itoa' function: -12345
The function 'itoa' Invoked!
Result: -12345

Source string: Hello, world!
Substring: world
The function 'strindex' Invoked!
Substring found at index: 7
The function 'getop' Invoked!
Enter an expression: 65 - 98 - 197
Operand or operator: 65
Operand or operator: -
Operand or operator: 98
Operand or operator: -
Operand or operator: 197
Operand or operator:

^Z

--------------------------------
Process exited after 23.17 seconds with return value 0
Press any key to continue . . .
```