UCS 2201 Fundamentals and Practice of Software Development
A6: Programming using Strings

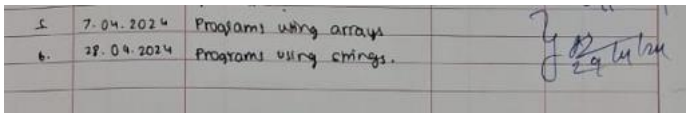
Batch 2023-2027
Name: SK Shivaanee
Register ID: 2310257
Section: CSE B

Dr. Chitra Babu, Dr. D. Thenmozhi, Dr. M. Saritha

Learning Outcome:
You will be able to implement string operations in C with the following features:
● Using built-in and user-defined string functions
● Passing strings to a function

You will be able to adapt to the following best practices
● Modular programming
● To develop code incrementally

Assignment: Write the algorithm and solve the following problems by implementing in C. (CO7, K3, 1.3.1, 1.4.1, 2.1.2, 2.1.3, 2.4.3, 3.2.2, 3.4.3, 4.1.2, 4.2.1, 5.2.2, 13.2.1, 13.3.2, 13.4.2, 14.2.1, 14.2.2)



1. Read a paragraph with multiple sentences that are separated by ".". Write a program to find the sentence with the maximum number of words. Apply the following constraints.
   ○ Do not use any built-in string functions. The required string functions need to be defined.
   ○ If more than one sentence is with the maximum number of words, display the first sentence with the maximum words.
   Example:
   Input: C is a general-purpose computer programming language. It was created in the 1970s by Dennis Ritchie. It remains very widely used and influential.
   Output: Sentence with maximum number of words: It was created in the 1970s by Dennis Ritchie. Number of words: 9

for function count_words:

1. Start count_words function with a parameter sentence.
2. Initialize an integer variable words to 0 to store the count of words.
3. Initialize an integer variable i to 0 to use as an index for iterating through the characters of the sentence.
4. Iterate through each character of the sentence until the null terminator
   a. If the current character is a space
   b. Increment the index i by 1 to move to the next character.
5. Add 1 to the words count to account for the last word.
6. Return the value of the words count.
7. End

For function find_max_sentence

1. Start
2. Initialize a character array max_sentence to store the sentence with the maximum number of words
3. Initialize an integer variable max_words to 0 to store the maximum number of words found
4. Initialize a character array current_sentence to store the current sentence being processed.
5. Initialize an integer variable current_words to 0 to store the number of words in the current sentence
6. Initialize an integer variable i to 0 to use as an index for iterating through the characters of the text.
7. Initialize an integer variable j to 0 to use as an index for building the current_sentence array
8. Iterate through each character of the text until the null terminator is encountered.
   a. If the current character is a period '.' indicating the end of a sentence
      (i) Null terminate the current_sentence array.
      (ii) Count the number of words in the current sentence using the count_words function.
      (iii) If the number of words in the current sentence is greater than max_words:
         - Update max_words with the current number of words.
         - Copy the current sentence to the max_sentence array. ustrcpy
9. Print the sentence
10. Stop.

```c
#include <stdio.h>
// Function to count words in a sentence
int count_words(char *sentence) {
    int words = 0;
    int i = 0;
    while (sentence[i] != '\0') {
        if (sentence[i] == ' ') {
            words++;
        }
        i++;
    }
    // Add 1 to account for the last word
    return words + 1;
}
// Function to find the sentence with the maximum number of words
void find_max_sentence(char *text) {
    char max_sentence[1000];
    int max_words = 0;
    char current_sentence[1000];
    int current_words = 0;
    int i = 0;
    int j = 0;
    while (text[i] != '\0') {
        if (text[i] == '.') {
            current_sentence[j] = '\0'; // Null terminate the string
            current_words = count_words(current_sentence);
            if (current_words > max_words) {
                max_words = current_words;
```

```c
                strcpy(max_sentence, current_sentence);
            }
            // Reset current sentence
            j = 0;
        } else {
            current_sentence[j] = text[i];
            j++;
        }
        i++;
    }
    printf("Sentence with maximum number of words: %s\n", max_sentence);
    printf("Number of words: %d\n", max_words);
}
int main() {
    char text[] = "C is a general-purpose computer programming language.It
        was created in the 1970s by Dennis Ritchie.It remains very widely
        used and influential.";
    find_max_sentence(text);
    return 0;
}
```

```
Sentence with maximum number of words: It was created in the 1970s by Dennis
    Ritchie
Number of words: 9
```

2. Write a program which replaces all the occurrences of a substring with another in a given line of text.
   Example:
   Input: “There are 30 bananas on a banana-tree”, “banana”, “apple”
   Output: “There are 30 apples on a apple-tree”.

Algorithm:

For function replaceSubstring:

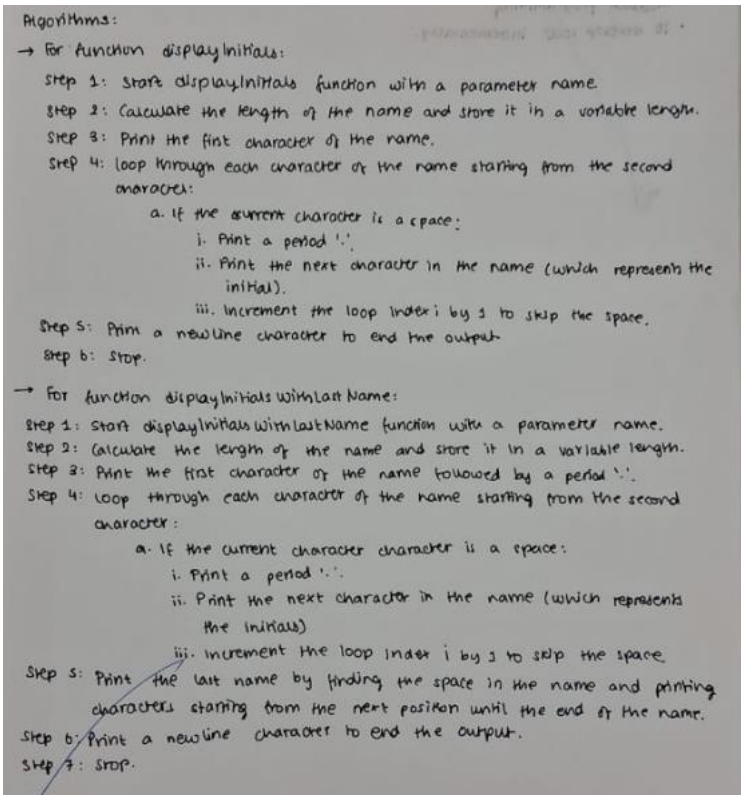
1. Start replaceSubstring function with parameters text, oldSubstr, and newSubstr.
2. Calculate the lengths of the oldSubstr, newSubstr, and text, and store them in variables oldLen, newLen, and textLen, respectively.
3. Initialize a character array newText to store the modified text.
4. Initialize an integer variable newTextIndex to 0 to keep track of the current index
5. Loop through each character of the text (indexed by variable i) until the end of the text is reached:
   a. Check if the substring starting at the current position (text[i] to text[i+oldLen-1]) matches the oldSubstr using strncmp.
   b. If the substring matches oldSubstr:
      i. Copy each character of newSubstr into the newText array, starting from newTextIndex, to replace the old substring.
      ii. Increment newTextIndex by the length of newSubstr.
      iii. Skip the old substring in the original text by incrementing i by oldLen-1.
   c. If the substring does not match oldSubstr:
      i. Copy the current character from the original text (text[i]) into the newText array at position newTextIndex.
      ii. Increment newTextIndex by 1.
6. Null-terminate the newText array by assigning '\0' to newText[newTextIndex].
7. Print the modified text stored in newText.
8. Stop.

```c
#include <stdio.h>
#include <string.h>
// Function to replace all occurrences of a substring with another substring
void replaceSubstring(char *text, char *oldSubstr, char *newSubstr) {
    int oldLen = strlen(oldSubstr);
    int newLen = strlen(newSubstr);
    int textLen = strlen(text);
    // Buffer to store the modified text
    char newText[1000];
    int newTextIndex = 0;
    // Loop through the text
    for (int i = 0; i < textLen; i++) {
        // Check if the current position matches the old substring
        if (strncmp(&text[i], oldSubstr, oldLen) == 0) {
            // Copy the new substring into the new text
            for (int j = 0; j < newLen; j++) {
                newText[newTextIndex] = newSubstr[j];
                newTextIndex++;
            }
            // Skip the old substring in the original text
            i += oldLen - 1;
        } else {
            // Copy the current character from the original text
            newText[newTextIndex] = text[i];
            newTextIndex++;
        }
    }
    // Null-terminate the new text
    newText[newTextIndex] = '\0';
    // Print the modified text
    printf("Output: %s\n", newText);
}
int main() {
    char text[] = "There are 30 bananas on a banana-tree";
    char oldSubstr[] = "banana";
    char newSubstr[] = "apple";

    replaceSubstring(text, oldSubstr, newSubstr);
    return 0;
}
```

```
Output: There are 30 apples on a apple-tree
```

3. Write a program to read a given name and then display it in the following abbreviated forms a. Input: Janak Raj Thareja Output: JRT b. Input: Janak Raj Thareja Output: J. R. Thareja



Algorithms:

→ For function displayInitials:

Step 1: Start displayInitials function with a parameter name.
Step 2: Calculate the length of the name and store it in a variable length.
Step 3: Print the first character of the name.
Step 4: loop through each character of the name starting from the second character:
  a. If the current character is a space:
    i. Print a period '.'.
    ii. Print the next character in the name (which represents the initial).
    iii. Increment the loop index i by 1 to skip the space.
Step 5: Print a newline character to end the output.
Step 6: Stop.

→ For function displayInitials WithLastName:

Step 1: Start displayInitials WithLastName function with a parameter name.
Step 2: Calculate the length of the name and store it in a variable length.
Step 3: Print the first character of the name followed by a period '.'.
Step 4: loop through each character of the name starting from the second character:
  a. If the current character character is a space:
    i. Print a period '.'.
    ii. Print the next character in the name (which represents the initials)
    iii. Increment the loop index i by 1 to skip the space.
Step 5: Print the last name by finding the space in the name and printing characters starting from the next position until the end of the name.
Step 6: Print a newline character to end the output.
Step 7: Stop.

```c
#include <stdio.h>
#include <string.h>
// Function to display name in the form of initials
void displayInitials(char *name) {
    int length = strlen(name);
    printf("Output: %c", name[0]); // Print the first character
    for (int i = 1; i < length; i++) {
        if (name[i] == ' ') {
            printf(". %c", name[i + 1]); // Print the initial after a space
            i++; // Skip the space
        }
    }
    printf("\n");
}
// Function to display name in the form of initials with last name
void displayInitialsWithLastName(char *name) {
    int length = strlen(name);
    printf("Output: %c. ", name[0]); // Print the first character with a
        period
    for (int i = 1; i < length; i++) {
        if (name[i] == ' ') {
            printf("%c. ", name[i + 1]); // Print the initial after a space
                with a period
            i++; // Skip the space
        }
    }
    // Print the last name
    for (int i = 0; i < length; i++) {
```

```c
    // Print the last name
    for (int i = 0; i < length; i++) {
        if (name[i] == ' ') {
            printf("%s\n", &name[i + 1]);
            break;
        }
    }
}
int main() {
    char name[] = "Janak Raj Thareja";
    // Display name in the form of initials (JRT)
    displayInitials(name);
    // Display name in the form of initials with last name (J. R. Thareja)
    displayInitialsWithLastName(name);
    return 0;
}
```

```
Output: J. R. T
Output: J. R. T. Raj Thareja
```