Informatics II
Exercise 1

Published date: March 01, 2021
Labs date: Week 2

Introduction to C

Task 1

Solution : There are two steps in the solution:

- **Calculate the length of the string.** Iterate each character of the string from the beginning of the string, using an index, until the null character($\backslash 0$). Once you find the null character, stop iterating the string and the value of index is the length of the string.
- **Reverse the string.** Create a new array for characters first, and then iterate each character of the string, and place each character to the new array (but start from the end of the array). Once we find the null character, stop iterating the string and the new array will be the reversed string.

Pseudocode

Algo: REVERSESTRING(s)

$strlen \leftarrow 0$;
while $s[i] \neq$ '$\backslash 0$' **do**
  $strlen = strlen + 1$
while $s[i] \neq$ '$\backslash 0$' **do**
  $reversed[strlen - i - 1] \leftarrow s[i]$
return reversed

Task 2

Solution: The most straightforward solution is iterating i from 1 to the given integer n, and check if i^2 equals to n. If there is such an i, break the for-loop and return *True*. Otherwise if there is no such an i until n, return *False*.

Pseudocode

Algo: ISPERFECTSQUARE(n)

for $i = 1$ **to** n **do**
  if $i^2 == n$ **then**
    print "$TRUE$"
else
  print "$FALSE$";

Improvement: In fact, you do not need to iterate from 1 to n. Instead, iterating from 1 to $\sqrt{n}$ is enough to determine if it is perfect square. In another word, for each *i* in our iteration, we check if $i^2 \leq n$. Then there are two cases:

- If $i^2 \leq n$, the function further checks if $i^2 == n$. If so, the given number is a perfect square number and the function should return *True*.
- If $i^2 > n$, then the perfect square number is not possible to be a perfect square number anymore. The function should return *False* in this case.

Pseudocode

Algo: isPerfectSquare(n)

while $i^2 \leq n$ **do**
  if $i^2 == n$ **then**
    print "$TRUE$"
print "$FALSE$";

Task 3

Solution: You will need nested loops to iterate over two-dimensional array to solve this task. The result matrix is a known to be a 3×3 matrix, hence, in the outer loop, you calculate the entries in every row of the result. Then in the inner loop, with the known row, you calculate the entry in this row column by column. Then the task now is to calculate the entry at a known row r and the column c. To calculate this, you iterate over the row in the first matrix and the column in the second matrix at the same time, multiply these two values and sum them.

Pseudocode

Algo: multiplyMatrix(A,B)

$result \leftarrow 3 \times 3$ empty array;
for $i = 0$ **to** *3* **do**
  for $j = 0$ **to** *3* **do**
    $sum \leftarrow 0$
    for $k = 0$ **to** *3* **do**
      $sum \leftarrow sum + A[i][k] \times B[k][j]$
    $result[i][j] = sum$

Sorting

Task 4

Solution: When performing selection sort in ascending order, you select the smallest item and swap it with the first item. But if you want to perform selection sort in descending order, you will need to select the largest item and swap it with the first item.

Algo: ascselectSort(A,n)

int $position, value$
for $i = 1$ **to** $n - 1$ **do**
  $min = i$ **for** $j = i + 1$ **to** n **do**
    if $A[j] < A[k]$ **then**
      $min_i d = j$
  exchange $A[i], A[min]$
return A

Algo: DESCSELECTSORT(A,n)

```
int position, value
for i = 1 to n − 1 do
    min = i for j = i + 1 to n do
        if A[j] > A[k] then
            min_i d = j
    exchange A[i], A[min]
return A
```