BITP 1113: PROGRAMMING TECHNIQUES
1BITS
Lab 8
Array Part 1

Exercise 1

Write a program to let a user enters a word. Then the program will verify the word is either palindrome or regular. A palindrome word is a sequence of characters that reads the same backward as forwarding such as noon and refer.

1. Analyse input, process and output.

Input : `word`
Process : a) Find the length of `word`.
b) Change the `word` to upper case.
c) Copy the `word` array into the `reverse` array in a reverse manner
d) Compare both arrays to check the palindrome status.
Output : `word` status

2. Create a new project named `Array1D`
3. At **Solution Explorer**, right-click **Source Files** folder and add new item named `Palindrome.cpp`.
4. Write the default code.
5. Declare array variables `word` and `reverse` with size declarator is 10. (Line 5 and 6).

```
4  int main() {
5      char word[10]; // declare character array to store the word (input)
6      char reverse[10]; // declare character array to store the reverse word
7      return 0;
8  }
```

6. Insert code to receive the value of `word` from a user (Line 7 and 8).

```
4  int main() {
5      char word[10]; // declare character array to store the word (input)
6      char reverse[10]; // declare character array to store the reverse word
7      cout << "Enter a word: ";
8      cin >> word; // input word
9      return 0;
10 }
```

7. Insert code to find the length of `word` (Line 12).

```
11     // Find the length of word
12     length = strlen(word);
```

8. Declare the variable `length` (Line 7).

```
4  int main() {
5      char word[10]; // declare character array to store the word (input)
6      char reverse[10]; // declare character array to store the reverse word
7      int length; // declare variable to store the array length
```

9. Insert code to change the `word` to upper case for standardising the characters. This step is executed to avoid logic error involves the same characters that are written in upper case and lower case such as ‘K’ and ‘k’ (Line 14 to 18).

```
14      // Change the word to upper case
15      for (int i = 0; i < length; i++) {
16          // update the element to upper case
17          word[i] = static_cast<char>(toupper(word[i]));
18      }
```

10. Insert code to copy the `word` in a reverse manner and store it in another array, i.e. `reverse` array (Line 20 to 25).

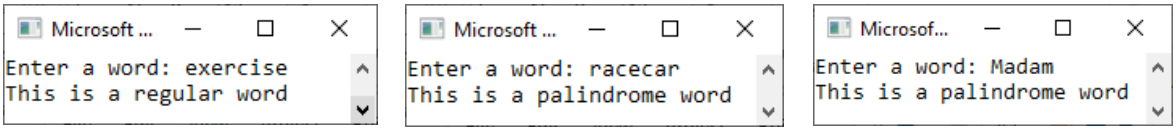
```
20      // Copy the word in reverse manner.
21      // Start read the word from the last to the first character.
22      // Start store the reverse word from subscript 0.
23      for (int fwd = length-1, bwd = 0; fwd >= 0; fwd--, bwd++) {
24          reverse[bwd] = word[fwd];
25      }
```

11. Insert code to compare both `word` and `reverse` arrays. The character-by-character comparison will be executed (Line 27 to 41).
    a. If the program found non-identical characters, it will stop comparing and display the status as a regular word.
    b. If not, the `for` statement will continue executing until the last character and display the status as a palindrome word.

```
27      // Compare both word and reverse arrays (character-by-character)
28      for (int i = 0; i < length; i++) {
29          // if both elements of the same subscript are not identical
30          if (word[i] != reverse[i])
31          {
32              cout << "This is a regular word\n"; // display the result
33              break; // exit the loop
34          }
35          // if comparison reach the last subscript
36          // this code only executed if both elements are identical
37          if (i == length-1)
38          cout << "This is a palindrome word\n"; // display the result
39      }
40      return 0;
41  }
```

12. Compile and run the program.

Example output



Exercise 2

Write a program to record three salespersons name and their sales for four months. Then the program will calculate and display the total sales based on the salesperson name.

1. Analyse input, process and output.

Input : 3 salespersons `name` and `sales` for four months

Process : calculate total sales,
`total += sales[subscript_salesperson][subscript_month];`

Output : `name` and `totalSales` for three salespersons.

2. At the same project, right-click **Source Files** folder at **Solution Explorer** and add new item named `TotalSales.cpp`. (You can create a new project instead of using the existing).
3. If you are using the same project, comment the `main()` function of **Exercise 1** or comment on the whole program to avoid the error of having more than one `main()` function in the same project.
4. Write the default code.
5. Declare the size declarator as constant because the program will store three salespersons name and four months of sales. The number of salespersons and months are fixed throughout the program.

```
1  #include <iostream>
2  using namespace std;
3
4  // declare the size declarator for array as constant
5  const int PERSON = 3, MONTH = 4;
```

6. Declare 1D string array variable `name` to store three salesperson `name` and 2D integer array variable `sales` to store sales of four months for each salesperson (Line 8 to 15).

```
7  int main() {
8      // Create Parallel Arrays
9      // to link the salesperson name and their sales
10     // Subscript 0 for 1D and row 0 for 2D represent first salesperson
11     // 2nd salesperson: Subscript 1 & Row 1
12     // 3rd salesperson: Subscript 2 & Row 2
13     // Columns of 2D array represent the months
14     string name[PERSON]; // declare 1D array. Store salesperson name
15     int sales[PERSON][MONTH]; // declare 2D array. Store salesperson sales
```

7. Include code to input elements which are salespersons `name` (1D) and `sales` (2D). Input `name` using single `for` loop and input `sales` using nested `for` loop (Line 18 to 27).

```
18     // Input elements for name and sales arrays
19     for (int row = 0; row < PERSON; row++) {
20         cout << "Salesperson Name: ";
21         getline(cin, name[row]); // receive name
22         for (int col = 0; col < MONTH; col++) {
23             cout << "Sales Month " << col + 1 << ": ";
24             cin >> sales[row][col]; // receive sales
25         }
26         cout << endl;
27     }
```

8. Insert code to calculate and display `name` and `total` sales (Line 29 to 38).

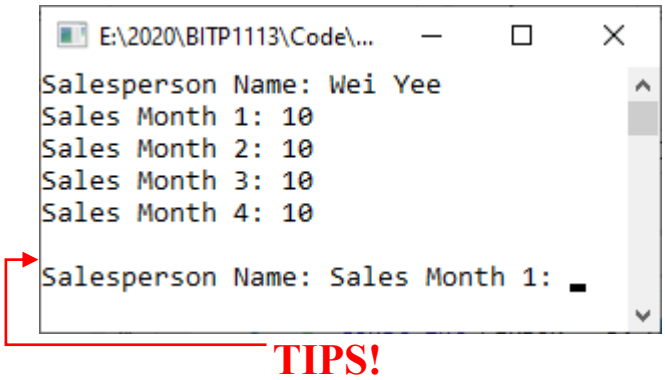
```
29    // Calculate and display total sales for each salesperson
30    cout << "------------SUMMARY-------------\n";
31    for (int row = 0; row < PERSON; row++) {
32        cout << "Total sales (" << name[row] << ") is " ; // display name
33        total = 0; // set total to zero for each salesperson
34        for (int col = 0; col < MONTH; col++) {
35            total += sales[row][col]; // calculate the total sales
36        }
37        cout << total << endl; // display total sales
38    }
39    return 0;
```

9. Declare variable `total` to fix the error (Line 17).

```
16    int sales[PERSON][MONTH]; // declare 2D array. Store
17    int total; // declare variable to store total sales
```

10. Compile and run the program.

Example output



- A user <u>could not input</u> a name for the second salesperson because the compiler jumps to <u>execute the next line of code</u> which is an instruction to input the sales.
  - This commonly happens when we used the `getline` statement.
- To overcome this, use `cin.ignore()` function to <u>ignore or clear</u> one or more characters from the <u>input buffer</u>.

11. Insert `cin.ignore()` function (Line 28 and 29).

```
19    // Input elements for name and sales arrays
20    for (int row = 0; row < PERSON; row++) {
21        cout << "Salesperson Name: ";
22        getline(cin, name[row]); // receive name
23        for (int col = 0; col < MONTH; col++) {
24            cout << "Sales Month " << col + 1 << ": ";
25            cin >> sales[row][col]; // receive sales
26        }
27        cout << endl;
28        // to clear one or more characters from the input buffer
29        cin.ignore();
30    }
```

12. Compile and run the program.

Example output

```
Select Microsoft Visu...
Salesperson Name: Ahmad Adi
Sales Month 1: 10
Sales Month 2: 10
Sales Month 3: 10
Sales Month 4: 10

Salesperson Name: Wei Yee
Sales Month 1: 20
Sales Month 2: 20
Sales Month 3: 20
Sales Month 4: 20

Salesperson Name: Suresh
Sales Month 1: 30
Sales Month 2: 30
Sales Month 3: 30
Sales Month 4: 30

------------SUMMARY-------------
Total sales (Ahmad Adi) is 40
Total sales (Wei Yee) is 80
Total sales (Suresh) is 120
```

Lab Attendance Week 10

Question

Write a program to print a graph that shows the number of students based on their grade. In order to illustrate this graph, the program will display the grade as a guideline to the user to enter the number of students. Use the values, 'A', 'B', 'C', 'D' and 'E' to initialise the grade variable then manipulate this variable when necessary. Complete the program to produce an output as shown in Figure 1.

```
Microsoft Visual Studio Debug Console
Enter number of grade A students: 5
Enter number of grade B students: 7
Enter number of grade C students: 11
Enter number of grade D students: 3
Enter number of grade E students: 1

-----GRAPH: NUMBER OF STUDENTS BASED ON GRADE-----
Grade A |  * * * * * 5
Grade B |  * * * * * * * 7
Grade C |  * * * * * * * * * * * 11
Grade D |  * * * 3
Grade E |  * 1
```

Figure 1: Example input/output:

Submit this exercise at ULearn before 12.00 p.m. 17 December 2020 (Thursday).

~Do something today that your future self will thank you for~
- Anonymous