GECG10069 (561085) F25: Introduction to Programming (C++)

Lab13: Dynamic Array & UDTs

What you will learn from Lab 13

In this laboratory, you will understand how to manipulate dynamic arrays and UDTs.

TASK 13-1 : DYNAMIC ARRAY

- ✔ Dynamic arrays allow various lengths of arrays.
- ✔ ***new*** operator (`new T ()`) is used to allocate a block of raw storage of *`sizeof(T)`*;
  `new T [n]` is used to allocate a block of raw storage of *`n * sizeof(T)`*
- ✔ ***delete*** operator is used to release the memory space allocated by ***new*** operator;
  `delete []` is used to release memory allocated by `new T[n]`
- ✔ Program `lab13-1` shows an example of using dynamic arrays.

```
//File: lab12-1.cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter the size of array: ";
    cin >> n;

    int *vec = new int[n];
    for (int idx = 0; idx < n; idx++)
    {
        vec[idx] = idx;
    }

    for (int idx = 0; idx < n; idx++)
    {
        cout << vec[idx] << " ";
    }
    cout << endl;

    delete []vec;

    return 0;
}
```

- ➢ It is essential to release the memory space by ***delete*** operator.
- ➢ `new T ()` is used to allocate a single object;
  `delete` is used to release the memory allocated by `new T()`.
- ➢ `new T[n]` is used to allocate an object array;
  `delete []` is used to release the space allocated by `new T[n]`.

TASK 13-2 : ENUM

✔ Please execute this program `lab13-2` and compare with your prediction.

```
//File: lab13-2.cpp
#include <iostream>
using namespace std;

enum baseball {FIRSTBAT = 1, SECONDBAT = 2, THIRDBAT = 3, FOURTHBAT = 4};

int main()
{
    baseball BatOrder = baseball(2);
    if (BatOrder == THIRDBAT)
    {
        cout << "The BatOrder is THIRDBAT" << endl;
    }

    return 0;
}
```

- Try to modify the program if there is no output on screen.

TASK 13-3 : STRUCTURE

✔ Please execute this program `lab13-3` and compare with your prediction.

```
//File: lab13-3.cpp
#include <iostream>
using namespace std;

struct Time
{
    int hours;
    int minutes;
    int seconds;
};

Time SetTime()
{
    Time now;
    cout << "Please Enter Current Time:" << endl;
    cout << "Current hour:";
    cin   >> now.hours;
    cout << "Current minute:";
    cin >> now.minutes;
    cout << "Current second:";
    cin >> now.seconds;

    return now;
}

void ShowSeconds(const Time &now)
{
    int seconds = 0;
    seconds = now.hours * 3600 + now.minutes * 60 + now.seconds;
    cout << "Total seconds of today:  " << seconds << endl;

    now.hours = 12;
    seconds = now.hours * 3600 + now.minutes * 60 + now.seconds;
    cout << "Total seconds of new:  " << seconds << endl;

    return ;
```

```
}
int main()
{
    Time t = {0, 0, 0};
    t = SetTime();

    ShowSeconds(t);

    return 0;
}
```

EXERCISE 13-1 : MATRIX ADDITION

Description -

Write a C++ program that:

1. Reads two integers **m** and **n** (both > 0), representing the dimensions of a matrix.
2. Dynamically allocates **two** m × n integer matrices using a **double pointer** (int**)
3. Reads the elements of **Matrix A** and **Matrix B** from the user.
4. Computes their element-wise sum and stores the result in a new dynamically allocated matrix **Matrix C**.
5. Prints the resulting matrix **C** in m rows and n columns.
6. Correctly frees **all dynamically allocated memory** using delete[] in the proper order.

You must complete the functions

```
int** CreateMatrix(int m, int n);
void DeleteMatrix(int** mat, int m);
```

CreateMatrix should:

- Dynamically allocate an m × n matrix

- Return the pointer to the matrix

DeleteMatrix should:

- Correctly free all rows
- Free the top-level pointer array

Sample Test Cases -

Sample Input - 1
2 3 1 2 3 4 5 6 6 5 4 3 2 1
Sample Output - 1
7 7 7 7 7 7

Sample Input - 2
3 2 1 0 -1 2 4 5 3 1 2 2 1 1
Sample Output - 2
4 1 1 4 5 6

EXERCISE 13-2 : SMARTEST STUDENT

Description -

Write a C++ function to find the student who gets the highest score.

1. Defines a struct for storing student information:
```
struct Student{
    char name[20];
    int score;
};
```
2. Reads an integer `N (number of students, 0 < N ≤ 50).`
3. Dynamically allocates an array of Student:
```
Student* list = new Student[N];
```
4. Reads N students' data from input. Each student has a name (no spaces) and an integer score.
5. Implements a function:
```
Student* findTopStudent(Student* list, int N);
```

This function should:

- Go through the array using either array indexing (list[i]) or pointer arithmetic ((list + i)->score)
- Find the student with the highest score
- If multiple students have the same highest score, return the first one appearing in the list
- Return a pointer to that Student
- If N is 0 (no students), return nullptr

In main():

- Call findTopStudent(list, N)
- If the returned pointer is not nullptr, print the student's name who gets the highest score
- Correctly free the dynamically allocated memory

Sample Testcases -

Sample Input - 1
3 Alice 75 Bob 92 Carol 88
Sample Output - 1
Top student: Bob

Sample Input - 2
5 Tom 60 Jerry 60 Maya 100 Ken 90 Lily 85
Sample Output - 2
Top student: Maya