Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи №1 з дисципліни
“Програмування 2. Структури даних та алгоритми”

# “ДОСЛІДЖЕННЯ ОПИСУ ТА ВИКОРИСТАННЯ КЛАСІВ”

Варіант №13

Виконав студент Флорчук Назарій Петрович

Перевірив викладач Проскура Світлана Леонідівна

Мета роботи:

1. Вивчити устрій і правила написання класу, механізми створення, використання та нищення об’єктів класів мовами програмування C++ та C#.

Завдання:

1. Вивчити та дослідити як в мовах програмування C++ та C# створювати класи з членами – даними інших класів.
2. Написати програми мовами програмування C++ та C#, що демонструють випадки створення класів-контейнерів з об’єктів інших класів згідно з варіантом.

| Опис класів | Завдання |
|---|---|
| Клас – рядок, який містить методи, необхідні для роботи методів класу-контейнера. Клас – контейнер, який є абстракцією тексту та складається з об’єктів класу – рядка та методів додавання рядка до тексту, видалення рядка з тексту, очищення тексту, повернення найбільшого рядка, повернення відсотка символів-цифр у тексті, повернення загальної кількості символів. | Створити один чи декілька об’єктів – текстів та продемонструвати їх функціональність, застосовуючи операції над текстом. |

Код програми (C++):

```
#include <iostream>
#include <vector>
#include <regex>

class Row
{
   private:
      std::string _m_row;

   public:
      Row(std::string row = "")
      {
         this->_m_row = row;
      }

      /**
       * Get Row string
       */
      std::string Get()
      {
         return this->_m_row;
      }

      /**
       * Get Row length
       */
      unsigned int Length()
      {
         return this->_m_row.length();
      }
};

class Text
{
   private:
      std::vector<Row> _m_rows;

   public:
      Text(Row rows[], unsigned int size)
      {
         for (unsigned int i = 0; i < size; i++)
         {
            this->_m_rows.push_back(rows[i]);
         }
      }

      /**
```

```
 * Add Row to the Text
 */
void Add(Row row)
{
   this->_m_rows.push_back(row);
}

/**
 * Delete Row from The Text
 */
void Del(Row row)
{
   for (unsigned int i = this->_m_rows.size() - 1; i >= 0; i--)
   {
      if (this->_m_rows[i].Get() == row.Get())
      {
         this->_m_rows.erase(this->_m_rows.begin() + i);

         break;
      }
   }
}

/**
 * Clear Text from the Rows
 */
void Clear()
{
   this->_m_rows.clear();
}

/**
 * Get the Text length (sum of all the Rows length)
 */
unsigned int Length()
{
   unsigned int length = 0;

   for (Row row : this->_m_rows)
   {
      length += row.Length();
   }

   return length;
}

/**
 * Get longest Text Row
 */
```

```
Row Longest()
{
   if (this->_m_rows.size() == 0)
   {
      return Row();
   }

   unsigned int index = 0;
   unsigned int length = 0;

   for (unsigned int i = 0; i < this->_m_rows.size(); i++)
   {
      if (this->_m_rows[i].Length() >= length)
      {
         index = i;
         length = this->_m_rows[i].Length();
      }
   }

   return this->_m_rows[index];
}

/**
 * Get percentage of digits in Text Rows
 */
double Percentage()
{
   double count = 0.0;
   double length = 0.0;

   std::regex regex("[0-9]");

   for (Row row : this->_m_rows)
   {
      length += row.Length();

      std::string row_string = row.Get();

      auto digits_begin = std::sregex_iterator(row_string.begin(), row_string.end(), regex);
      auto digits_end = std::sregex_iterator();

      count += std::distance(digits_begin, digits_end);
   }

   return count / (length / 100.0);
}

/**
 * Print out Text Rows to the console
```

```
        */
        void Print()
        {
            for (Row row : this->_m_rows)
            {
                std::cout << row.Get() << std::endl;
            }
        }

        ~Text()
        {
            this->_m_rows.clear();
        }
};

int main()
{
    Row rows[] = {Row("First row."), Row("Second row.")};

    Text text(rows, 2);

    std::cout << "____ Initial Text object:" << std::endl;

    text.Print();

    Row forDeletionPurposesRow("For deletion purposes Row.");

    text.Add(forDeletionPurposesRow);
    text.Add(Row("Third row."));

    std::cout << "____ Text object (with additional two Row objects):" << std::endl;

    text.Print();

    text.Del(forDeletionPurposesRow);

    std::cout << "____ Text object (after deletion Row):" << std::endl;

    text.Print();

    std::cout << "____ Text object longest Row:" << std::endl;
    std::cout << text.Longest().Get() << std::endl;

    text.Clear();

    std::cout << "____ Text object after clean up:" << std::endl;

    text.Print();
```

```
    text.Add(Row("1-st row."));
    text.Add(Row("Last (2-nd) row."));

    std::cout << "____ Text object (with additional two Row objects):" << std::endl;

    text.Print();

    std::cout << "____ Text object length:" << std::endl;
    std::cout << text.Length() << std::endl;

    std::cout << "____ Text object percentage of digits in Rows:" << std::endl;
    std::cout << text.Percentage() << std::endl;

    exit(0);
}
```

```
    @debian:~/Documents/kpi/basics_of_programming_2/lab_1$ g++ lab_1.cpp -o lab_1
    @debian:~/Documents/kpi/basics_of_programming_2/lab_1$ ./lab_1
____ Initial Text object:
First row.
Second row.
____ Text object (with additional two Row objects):
First row.
Second row.
For deletion purposes Row.
Third row.
____ Text object (after deletion Row):
First row.
Second row.
Third row.
____ Text object longest Row:
Second row.
____ Text object after clean up:
____ Text object (with additional two Row objects):
1-st row.
Last (2-nd) row.
____ Text object length:
25
____ Text object percentage of digits in Rows:
8
```

Код програми (C#):

```
using System.Collections.Generic;
using System.Text.RegularExpressions;

class Row
{
    private string _m_row;

    public Row(string row = "")
    {
        this._m_row = row;
    }

    /**
     * Get Row string
     */
    public string Get()
    {
        return this._m_row;
    }

    /**
     * Get Row length
     */
    public int Length()
    {
        return this._m_row.Length;
    }
}

class Text
{
    private List<Row> _m_rows = new List<Row>();

    public Text(Row[] rows)
    {
        this._m_rows.AddRange(rows);
    }

    /**
     * Add Row to the Text
     */
    public void Add(Row row)
    {
        this._m_rows.Add(row);
    }

    /**
```

```
 * Delete Row from The Text
 */
public void Del(Row row)
{
   this._m_rows.Remove(row);
}

/**
 * Clear Text from the Rows
 */
public void Clear()
{
   this._m_rows.Clear();
}

/**
 * Get the Text length (sum of all the Rows length)
 */
public int Length()
{
   int length = 0;

   foreach (Row row in this._m_rows)
   {
      length += row.Length();
   }

   return length;
}

/**
 * Get longest Text Row
 */
public Row Longest()
{
   if (this._m_rows.Count == 0)
   {
      return new Row();
   }

   int index = 0;
   int length = 0;

   for (int i = 0; i < this._m_rows.Count; i++)
   {
      if (this._m_rows[i].Length() >= length)
      {
         index = i;
         length = this._m_rows[i].Length();
```

```
            }
        }

        return this._m_rows[index];
    }

    /**
     * Get percentage of digits in Text Rows
     */
    public double Percentage()
    {
        double count = 0.0;
        double length = 0.0;

        Regex regex = new Regex("[0-9]");

        foreach (Row row in this._m_rows)
        {
            length += row.Length();

            count += regex.Matches(row.Get()).Count;
        }

        return count / (length / 100.0);
    }

    /**
     * Print out Text Rows to the console
     */
    public void Print()
    {
        foreach (Row row in this._m_rows)
        {
            System.Console.WriteLine(row.Get());
        }
    }

    ~Text()
    {
        this._m_rows.Clear();
    }
}

class Application
{
    static void Main(string[] args)
    {
        Text text = new Text(new Row[] {new Row("First row."), new Row("Second row.")});
```

```
        System.Console.WriteLine("____ Initial Text object:");

        text.Print();

        Row forDeletionPurposesRow = new Row("For deletion purposes Row.");

        text.Add(forDeletionPurposesRow);
        text.Add(new Row("Third row."));

        System.Console.WriteLine("____ Text object (with additional two Row objects):");

        text.Print();

        text.Del(forDeletionPurposesRow);

        System.Console.WriteLine("____ Text object (after deletion Row):");

        text.Print();

        System.Console.WriteLine("____ Text object longest Row:");
        System.Console.WriteLine(text.Longest().Get());

        text.Clear();

        System.Console.WriteLine("____ Text object after clean up:");

        text.Print();

        text.Add(new Row("1-st row."));
        text.Add(new Row("Last (2-nd) row."));

        System.Console.WriteLine("____ Text object (with additional two Row objects):");

        text.Print();

        System.Console.WriteLine("____ Text object length:");
        System.Console.WriteLine(text.Length());

        System.Console.WriteLine("____ Text object percentage of digits in Rows:");

        System.Console.WriteLine(text.Percentage());
    }
}
```

```
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/lab_1$ mcs lab_1.cs
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/lab_1$ mono lab_1.exe
____ Initial Text object:
First row.
Second row.
____ Text object (with additional two Row objects):
First row.
Second row.
For deletion purposes Row.
Third row.
____ Text object (after deletion Row):
First row.
Second row.
Third row.
____ Text object longest Row:
Second row.
____ Text object after clean up:
____ Text object (with additional two Row objects):
1-st row.
Last (2-nd) row.
____ Text object length:
25
____ Text object percentage of digits in Rows:
8
```

Висновки / Відповіді на контрольні запитання:

1. Що таке клас? Що відрізняє клас і об’єкт?
   - Клас – набір властивостей (полів) та методів спільного призначення.
   - Об’єкт – екземпляр класу.
   - Клас – це тип даних, а об’єкт – це змінна типу клас.
   - Для класу не виконується виділення пам’яті, на відміну від об’єкту даного класу.
2. З яких частин складається опис класу?
   Задання класу починається із ключового слова **class**, після якщо задається ім’я даного класу. У тілі класу задаються його властивості (поля), методи класу, конструктор та деструктор класу.
3. Що таке конструктор та деструктор?
   Конструктор – метод класу, виклик якого виконується при кожному створенні екземпляру даного класу.
   Деструктор – метод класу, виклик якого виконується Garbage collector’ом, як тільки об’єкт класу завершив своє використання у програмі.
4. Наведіть приклади типів конструкторів.
   Дивитись у коді вище.
5. Для чого використовується конструктор?
   Для виділення пам’яті для об’єкту класу, при створенні його екземпляра, та для передачі певних аргументів класу, при створенні його екземпляру,
6. Для чого використовується деструктор?
   Для звільнення пам’яті використовуваної екземпляром класу, після остаточного використання об’єкту класу. Також для операцій, які необхідно виконати після остаточного використання об’єкту класу. Наприклад, якщо у нас є об’єкт класу з’єднання з базою даних (тобто при створенні екземпляру класу, відкривається нове підключення до бази даних), то після використання цього об’єкту, нам потрібний деструктор, який буде закривати це підключення до бази даних, коли воно нам уже не потрібне.
7. Наведіть приклади використання методів.
   Дивитись у коді вище.
8. Що таке контейнерний клас?
   Клас — контейнер — клас, призначений для зберігання і організації екземплярів певного класу.
9. Який порядок створення об’єкта контейнерного класу?
   Проводиться аналіз внутрішньої структури системи (коду), для виявлення самодостатніх підсистем (частин коду), які можна винести в окрему систему (клас), для подальшого їх використання іншими системами.
10. Який порядок видалення об’єкта контейнерного класу?
    Контейнери, в яких зберігаються значення певних об’єктів, відповідальні за видалення (знищення) цих об’єктів.
    Контейнери, в яких зберігаються посилання на значення певних об’єктів, не відповідальні за видалення (знищення) цих об’єктів.