Лабораторная работа № 2

Тема: Изучение базовых приемов работы с классами

Цель:

1. Закрепление навыков работы с классами
2. Закрепление навыков работы с динамической памятью на «куче»
3. Закрепление навыков работы с массивами

Порядок выполнения работы

- Ознакомиться с теоретическим материалом.
- Получить у преподавателя вариант задания.
- Реализовать задание своего варианта в соответствии с поставленными требованиями.
- Написать Unit-тесты с использованием Google Test.
- Создать репозиторий на GitHub.
- Отправить файлы лабораторной работы в репозиторий.
- Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы преподавателя (как из числа контрольных, так и по реализации программы).

Требования к программе

Используя в качестве образца класс **Array** (см. ниже), реализовать динамические контейнеры с использованием динамического массива.

- Каждый класс должен быть разделен на интерфейс и реализацию.
- Самостоятельно определить необходимые типы, поля и дополнительные методы.
- Реализовать генерацию исключений в конструкторах и методах при необходимости (использовать стандартные исключения).
- Реализовать арифметические операции: сложение, вычитание, копирование
- Реализовать операции сравнения: (больше, меньше, равно).
- Арифметические операции с присваиванием должны быть реализованы как методы класса.
- Перегрузку операторов применять не нужно
- Объекты классов должны быть иммутабельными (то есть не меняться после создания), результат методов должен возвращаться как новый экземпляр объекта

```
class Array
{
public:
   Array();
   Array(const size_t & n, unsigned char t = 0);
   Array(const std::initializer_list< unsigned char> &t);
   Array(const string &t);
   Array(const Array& other);
   Array(Array&& other) noexcept;
   virtual ~Array() noexcept;
};
```

Классы:

1. Создать класс `Decimal` для работы с беззнаковыми целыми десятичными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является десятичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).

2. Создать класс `Hex` для работы с беззнаковыми целыми шестнадцатеричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является шестнадцатеричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
3. Создать класс `Octal` для работы с беззнаковыми целыми восьмеричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является восьмеричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
4. Создать класс `Four` для работы с беззнаковыми целыми чертвертичными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является четвертичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
5. Создать класс `Three` для работы с беззнаковыми целыми троичными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является троичной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
6. Создать класс `Five` для работы с беззнаковыми целыми пятиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является пятеричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
7. Создать класс `Six` для работы с беззнаковыми целыми шестиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является шестиричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
8. Создать класс `Seven` для работы с беззнаковыми целыми семеричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый элемент которого является семиричной цифрой. Младшая цифра имеет меньший индекс (единицы — в нулевом элементе массива).
9. Создать класс `Eleven` для работы с беззнаковыми целыми одиннадцатиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является одиннадцатиричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
10. Создать класс `Twelve` для работы с беззнаковыми целыми двенадцатиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является двенадцатиричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
11. Создать класс `Thirteen` для работы с беззнаковыми целыми тринадцатиричными числами, используя для представления числа массив из элементов типа `unsigned char`, каждый из которых является тринадцатиричными цифрой. Младшая цифра имеет меньший индекс (единицы – в нулевом элементе массива).
12. Создать класс `Money` для работы с денежными суммами. Сумма должна быть представлена массивом из элементов типа `unsigned char`, каждый элемент которого – десятичная цифра. Младший индекс соответствует младшей цифре денежной суммы. Младшие две цифры — копейки.
13. Создать класс `Binary` для работы с двоичными беззнаковыми числами фиксированной длины. Число должно быть представлено массивом типа `unsigned char`, каждый элемент которого принимает значение 0 или 1. Младший бит имеет младший индекс.
14. Создать класс `BitString` для работы с битовыми строками. Битовая строка должна быть представлена массивом типа `unsigned char`, каждый элемент которого принимает значение 0 или 1. Должны быть реализованы все традиционные операции для работы с битовыми строками: `and`, `or`, `xor`, `not`.