Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №7
«Разработка простейшего класса на C++»
по курсу: «Языки и методы программирования»

Выполнил:
Студент группы ИУ9-21Б
Старовойтов А. И.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Целью данной работы является изучение базовых объектно-ориентированных возможностей языка C++.

Задачи

Выполнение лабораторной работы заключается в составлении на языке C++ программы, состоящей из трёх файлов:

- заголовочный файл declaration.h с объявлением одного из классов, приведённых в таблицах 1 – 16;
- файл implementation.cpp с определениями методов класса;
- файл main.cpp, содержащий функцию main и, возможно, вспомогательные функции для проверки работоспособности класса.

Реализация класса не должна опираться на стандартные контейнерные классы C++, то есть внутреннее состояние объектов класса должно быть реализовано через обычные массивы. Соответственно, в классе обязательно требуется реализовать:

- ☒ конструктор копий;
- ☒ деструктор (должен быть объявлен виртуальным);
- ☒ операцию присваивания.

Проверку работоспособности класса требуется организовать в функции main, размещённой в файле main.cpp. Проверка должна включать в себя:

- ☒ создание объекта класса в автоматической памяти;
- ☒ передачу объекта класса по значению в функцию;
- ☒ присваивание объекта класса переменной.

Полином с целочисленными коэффициентами, часть из которых может быть «неизвестна». Для полинома должны быть реализованы следующие операции:

- ☒ проверка, определён ли коэффициент полинома с индексом i;
- ☒ получение ссылки на i-тый коэффициент полинома (при этом неизвестному коэффициенту автоматически присваивается нулевое значение);

- ☒ дифференцирование с порождением нового полинома для хранения производной (коэффициенты производной, для вычисления которых нужно знать значения неизвестных коэффициентов полинома, становятся неизвестными).

Решение

polinom.h

```
#ifndef LAB6_POLINOM_H_
#define LAB6_POLINOM_H_

#include <cstdint>
#include <memory>
#include <optional>

class Polinom {
    public:
        using coefficient_type = std::int32_t;
        using data_type = std::optional<coefficient_type>;
        constexpr static data_type UNKNOWN = std::nullopt;

        Polinom();
        Polinom(std::initializer_list<data_type>);
        explicit Polinom(size_t);

        Polinom(const Polinom&);
        Polinom& operator=(const Polinom&);

        Polinom(Polinom&&) = default;
        Polinom& operator=(Polinom&&) = default;

        virtual ~Polinom() = default;

        std::string toString() const;
        size_t size() const;
        data_type* begin() const;
        data_type* end() const;

        bool isDefined(size_t i) const;
```

```
        coefficient_type& operator[](size_t i);
        Polinom differentiate() const;
    private:
        size_t sz = 1;
        std::unique_ptr<data_type[]> data =
         ↪  std::make_unique<data_type[]>(1);
};

#endif // LAB6_POLINOM_H_
```

polinom.cpp

```
#include "polinom.h"

#include <algorithm>
#include <sstream>
#include <string>

Polinom::Polinom() {
    data[0] = 0;
}

Polinom::Polinom(std::initializer_list<data_type> arr)
    : sz(arr.size()),
     ↪  data(std::make_unique<data_type[]>(sz)) {
        if (sz == 0) {
            sz = 1;
            data = std::make_unique<data_type[]>(1);
            data[0] = 0;
        }
        std::copy(arr.begin(), arr.end(), data.get());
    }

Polinom::Polinom(size_t n)
    : sz(n), data(std::make_unique<data_type[]>(sz)) {
        std::fill(this->begin(), this->end(), data_type{});
}

Polinom::Polinom(const Polinom& p)
```

```
    : sz(p.size()), data(std::make_unique<data_type[]>(sz))
     ↪  {
        std::copy(p.begin(), p.end(), this->begin());
}

Polinom& Polinom::operator=(const Polinom& p) {
    sz = p.size();
    data = std::make_unique<data_type[]>(sz);
    std::copy(p.begin(), p.end(), this->begin());
    return *this;
}

std::string Polinom::toString() const {
    std::stringstream res;
    std::uint32_t unknown_counter = 0;
    for (const auto& el : *this) {
        if (!el) {
            res << 'x' << std::to_string(unknown_counter) <<
 ↪  ' ';
            unknown_counter++;
            continue;
        }
        res << std::to_string(*el) << ' ';
    }
    return res.str();
}

Polinom::data_type* Polinom::begin() const {
    return data.get();
}

Polinom::data_type* Polinom::end() const {
    return data.get()+sz;
}

size_t Polinom::size() const {
    return sz;
}

bool Polinom::isDefined(size_t i) const {
```

```
    return data[i].has_value();
}

Polinom::coefficient_type& Polinom::operator[](size_t i) {
    if (!data[i]) {
        data[i] = 0;
    }
    return *data[i];
}

Polinom Polinom::differentiate() const {
    if (sz == 1) {
        return Polinom();
    }
    Polinom res(sz-1);
    for (size_t i = 1; i < sz; ++i) {
        if (!data[i]) {
            res.data[i-1] = UNKNOWN;
            continue;
        }
        res.data[i-1] = *data[i] * i;
    }
    return res;
}
```

main.cpp

```
#include "polinom.h"

#include <iostream>

int main() {
    {
        Polinom test({5, 3, 2, {}, Polinom::UNKNOWN, 2});
        std::cout << test.toString() << std::endl;
    }
    {
        Polinom test;
        std::cout << test.toString() << std::endl;
    }
```

```
    {
        Polinom test({5, 3, 2, {}, Polinom::UNKNOWN, 2});
        Polinom test2({1, 2, 3});
        test = test2;
        std::cout << test.toString() << std::endl;
        std::cout << test2.toString() << std::endl;
    }
    {
        auto func = [](Polinom test) {
            std::cout << test.toString() << std::endl;
        };
        Polinom test({5, 3, 2, {}, Polinom::UNKNOWN, 2});
        func(test);
    }
    {
        Polinom test({5, 3, 2, {}, Polinom::UNKNOWN, 2});
        std::cout << test.isDefined(0) << ' ' << test[0] <<
        ↪  std::endl;
        std::cout << test.isDefined(3) << ' ' << test[3] <<
        ↪  std::endl;
        std::cout << test.toString() << std::endl;
    }
    {
        Polinom test({5, 3, 2, {}, Polinom::UNKNOWN, 2});
        size_t end = test.size();
        for (size_t i = 0; i <= end; ++i) {
            std::cout << test.toString() << std::endl;
            test = test.differentiate();
        }
    }
    return 0;
}
```

Пример вывода

```
5 3 2 x0 x1 2

0

1 2 3
```

```
1 2 3

5 3 2 x0 x1 2

1 5
0 0
5 3 2 0 x0 2

5 3 2 x0 x1 2
3 4 x0 x1 10
4 x0 x1 40
x0 x1 120
x0 240
240
0
```