Курсовий проєкт «Interval»
Мова програмування C++

Неділя Ігор Ярославович[1]

[1]Спеціальність 111 «Математика», група «Комп'ютерна математика-1», механіко-математичний факультет Київського національного університету імені Тараса Шевченка

12 грудня 2023 р.

Анотація

У даній роботі розглядається програмна реалізація користувацьких структур «Interval» та «Square Inequality» мовою C++. Метою роботи є створення відповідних структур даних та реалізація методів, необхідних для їх коректної роботи, а також проведення тестування на коректність роботи даних структур. У рефераті детально описуються: алгоритм роботи програми, структури даних та можливості мови програмування C++, що використовуються, а також подається та аналізується лістинг коду. Окрема увага приділяється алгоритмам, що використовуються у програмі.

1 Вступ

Мета курсового проєкту: Мета курсового проєкту полягає в розробці програми, яка оперує структурами даних для представлення інтервалів, об'єднання і роботи з множинами інтервалів, а також розв'язання системи квадратних нерівностей за допомогою введених структур.

Поставимо задачі курсового проєкту:

1. **Опис структури ІНТЕРВАЛ:** Структура Interval має три поля - `type_interval`, `a` та `b`, які представляють тип інтервалу та його межі. За допомогою `type_interval` можна вказати, чи є межі плюс- або мінус-нескінченостями.

2. **Методи для структури** `Interval`:

   - **Введення/виведення:** дозволяє користувачу вводити та виводити значення інтервалу;
   - **Перетин:** реалізований як стандартне множення інтервалів;
   - **Об'єднання:** реалізоване як додавання інтервалів;
   - **Різниця інтервалів:** визначає інтервал, який включає всі значення одного інтервалу, але не включає значення іншого.

3. **Структура** `SetIntervals`: Ця структура є об'єднанням декількох інтервалів, що не перетинаються. Має поля - список (вектор, масив) інтервалів та кількість інтервалів.

4. **Методи для** `SetIntervals`:

   - **Перетин:** реалізований як множення списків інтервалів.
   - **Об'єднання:** реалізоване як додавання списків інтервалів.
   - **Різниця списків інтервалів:** визначає список інтервалів, який містить значення одного списку, але не містить значень іншого.
   - **Довжина списку:** сумарна довжина інтервалів у списку.
   - **Структура** `QuadraticInequality`: представляє квадратну нерівність з трьома дійсними числами та типом перерахування.
   - **Розв'язання системи квадратних нерівностей:** на виході отримується відповідь у вигляді `SetIntervals`, де кожен інтервал представляє розв'язок квадратної нерівності.

2 Опис програмних файлів

2.1 Опис головної програми

```
#include <iostream>
#include <string>
#include <square_inequality.hpp>

int main() {
    //

    std::cout << "Info: Equations are represented by 3
    coefficients (a, b, c) and a comparison operator ? (one of ==
    != > <) such that a * x^2 + b^x + c ? 0 evaluates to true\n";

    //

    std::cout << "Enter number of equations in system:\nn = " <<
    std::flush;
    unsigned n;
    std::cin >> n;

    //                                          ,
                                                          0
    if (n == 0) {
        return 0;
    }

    //
                               ,
    SquareInequality (                                          )
    std::vector<SquareInequality> equations;

    std::string op;

    //


    for (unsigned i = 0; i < n; i++) {
        double a, b, c;
        std::cout << "a b c = " << std::flush;
        std::cin >> a >> b >> c;

        std::cout << "operator ? = " << std::flush;
        std::cin >> op;

        //
```

```
34          EqualityKind eq =
35              op == "==" ? EqualityKind::Equal :
36              op == "!=" ? EqualityKind::NotEqual :
37              op == ">" ? EqualityKind::Greater : EqualityKind::Less
    ;
38
39          //                                          ,
    SquareInequality                        equations
40          equations.push_back(SquareInequality(a, b, c, eq));
41      }
42
43      //          ,
                                solution
44      IntervalSet solution = equations[0].solve();
45
46      //                  (              )          ,
                                                                ,


47      for (size_t i = 1; i < equations.size(); i++) {
48          solution = solution * equations[i].solve();
49      }
50
51      //                                              ,
52      std::cout << "Solution is:\n" << solution << std::endl;
53  }
```

Ця програма призначена для розв'язання системи квадратних нерівностей, які вводяться користувачем. Основний функціонал програми складається з таких кроків:

1. Користувачу виводиться повідомлення щодо представлення рівнянь у вигляді трьох коефіцієнтів `(a, b, c)` та оператора порівняння `?` (один із `==`, `!=`, `>`, `<`), що відповідає виразу `a * x^2 + b * x + c ? 0`, де `x` - змінна.

2. Користувач вводить кількість рівнянь `(n)`, що складають систему квадратних нерівностей.

3. Далі вводяться коефіцієнти кожного рівняння `(a, b, c)` та оператор порівняння для кожного з них.

4. Введені дані формуються у вектор об'єктів типу `SquareInequality`, який представляє кожне рівняння системи.

5. Програма виконує розв'язання кожного рівняння за допомогою методу `solve()` для кожного об'єкту `SquareInequality`.

6. Розв'язки об'єднуються у множину інтервалів за допомогою операції множення (`solution = solution * equations[i].solve()`).

7. Останній крок полягає у виведенні знайденого розв'язку у вигляді множини інтервалів на екран.

Отже, ця програма дозволяє користувачеві вводити систему квадратних нерівностей та отримувати їхні розв'язки у вигляді множини інтервалів.

3 Структура Interval

```
#include <iostream>
#include <cmath> // std::isinf
#include <interval.hpp>

Interval Interval::input_with_hint() {
    double l, r;
    std::cout << "Enter interval range:\nl = " << std::flush;
    std::cin >> l;
    std::cout << "r = " << std::flush;
    std::cin >> r;
    return Interval(l, r);
}

std::ostream& operator<<(std::ostream& out, const Interval&
    interval) {
    switch (interval.get_kind()) {
    case IntervalKind::EMPTY:
        out << "()";
        break;
    case IntervalKind::L_INFINITY:
        out << "(-inf, " << interval.get_r() << ")";
        break;
    case IntervalKind::R_INFINITY:
        out << "(" << interval.get_l() << ", +inf)";
        break;
    case IntervalKind::LR_INFINITY:
        out << "(-inf, +inf)";
        break;
    default:
        out << "(" << interval.get_l() << ", " << interval.get_r()
    << ")";
        break;
    }
    return out;
}

```

```
35 std::istream& operator>>(std::istream& in, Interval& interval) {
36     double l, r;
37     in >> l >> r;
38     interval = Interval(l, r);
39     return in;
40 }
41
42 // intersection
43 Interval operator*(const Interval& a, const Interval& b) {
44     double c_l = std::max(a.get_l(), b.get_l());
45     double c_r = std::min(a.get_r(), b.get_r());
46
47     // empty
48     if (std::fabs(c_r - c_l) < 1e-9) {
49         return {};
50     }
51
52     // new kind
53     auto c_kind =
54         std::isinf(c_l) && std::isinf(c_r) ? IntervalKind::
    LR_INFINITY :
55         std::isinf(c_l) ? IntervalKind::L_INFINITY :
56         std::isinf(c_r) ? IntervalKind::R_INFINITY : IntervalKind
    ::FINITY;
57
58     return Interval(c_l, c_r, c_kind);
59 }
```

1. Метод `input\_with\_hint()` Приймає введені користувачем значення для лівої та правої межі інтервалу та створює об'єкт типу `Interval` з цими значеннями. Повертає `Interval` з введеними користувачем значеннями.

2. Оператор виведення `operator<<`

   Залежно від типу інтервалу виводить його представлення у вигляді тексту. Якщо інтервал порожній, виводить "()". Якщо одна з меж - плюс чи мінус нескінченість, відповідно виводить діапазон у вигляді "(-inf, r)"або "(l, +inf)". Якщо обидві межі - плюс та мінус нескінченості, виводить "(-inf, +inf)". В інших випадках виводить діапазон у вигляді "(l, r)".

3. Оператор введення `operator>>`

   Отримує введені користувачем значення для лівої та правої межі інтервалу і створює об'єкт типу `Interval` з цими значеннями.

4. Оператор перетину `(operator*)`
   Приймає два інтервали та обчислює їх перетин. Знаходить максимальну

ліву межу (`c_l`) та мінімальну праву межу (`c_r`) між цими інтервалами. Якщо різниця між цими межами менша за допустиму похибку (1e-9), повертає порожній інтервал. Якщо обидві межі - плюс та мінус нескінченості, повертає інтервал з обома нескінченостями. Якщо тільки одна межа - плюс чи мінус нескінченість, повертає відповідний інтервал. В інших випадках повертає інтервал з обчисленими лівою та правою межами.