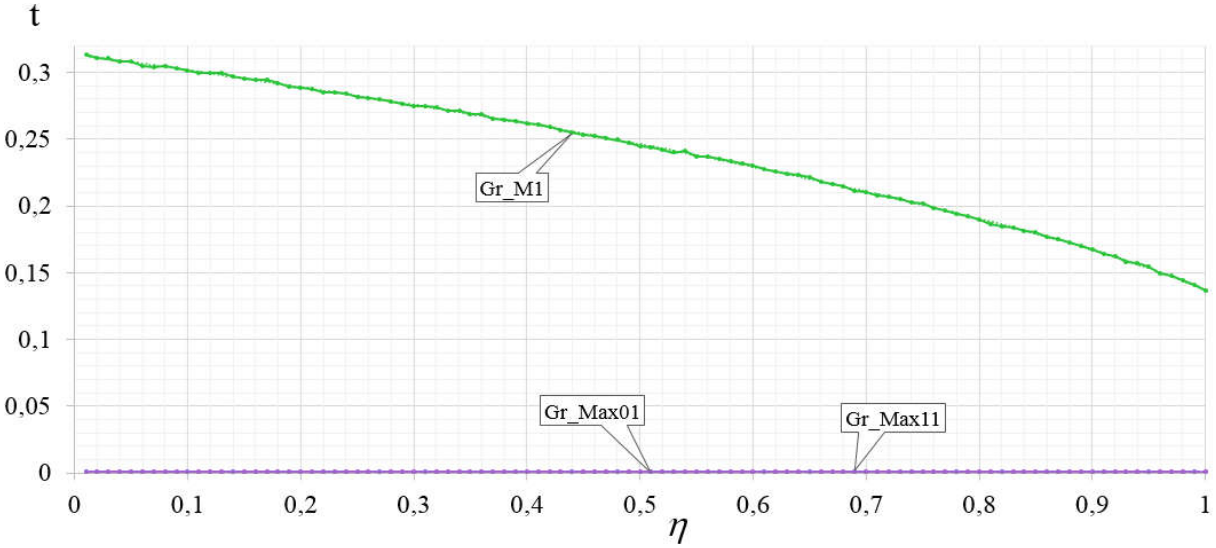

Рис. 2. Зависимость времени нахождения решения (в секундах) от параметра η

Для перебора по минимуму цветов временные затраты снижаются в ≈ 2,3 раза при использовании жадного выбора по сравнению со случайным. Дополнительная сортировка при переборе по степеням вершин не оказывает влияния ни на качество получаемых решений, ни на время их генерации.

Работа была частично поддержана РФФИ, грант № 17-07-00317-а.

Литература

1. Ватутин Э.И., Титов В.С., Емельянов С.Г. Основы дискретной комбинаторной оптимизации. М.: Аргамак-Медиа, 2016. 270 с.

2. Карпенко А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновлённые природой. М.: МГТУ им. Н.Э. Баумана, 2014. 446 с.

Л.В. Яблокова [1], Д.Е. Яблоков [2]

ПРИМЕНЕНИЕ ОБОБЩЁННОЙ КОНЦЕПЦИИ УНИВЕРСАЛЬНОГО ЧИСЛОВОГО ТИПА ДЛЯ РЕАЛИЗАЦИИ ОТКАЗОУСТОЙЧИВОГО АЛГОРИТМА ДЕЛЕНИЯ КОМПЛЕКСНЫХ ЧИСЕЛ

([1] Самарский национальный исследовательский университет имени академика С.П. Королева
[2] ООО «ИнтеллектСофт»)

Специалистам в области информационных технологий хорошо известно, что фундаментальной основой успешного проектного решения является выявление критериев общности, позволяющих сформировать концептуальные границы применяемого уровня абстракции. Причём для различных подходов эта часть процесса конструирования программного обеспечения имеет существенные отличия, что в основном связанно с качеством и эффективностью исполь-

зования имеющихся в распоряжении средств и методологий. В обобщённом программировании [1] основной акцент делается на проектировании таких алгоритмов и структур данных, которые бы работали в наиболее общей ситуации с наибольшим количеством встроенных или пользовательских типов. Смысловая составляющая обобщённой парадигмы выражается через понятие концепции [1], как средства описания семейств типов, объединенных синтаксической и семантической общностью.

Программирование с использованием обобщений, получившее своё развитие благодаря абстрактной математике [2], очень хорошо подходит для научных приложений из-за широкого применения в них развитого математического аппарата. Применяя обобщённый подход, основная идея которого состоит в использовании наиболее общих формулировок, возможно получение полного набора элементарных операций, которых вполне достаточно для реализации требуемых математических действий (Рис. 1).

```
1  interface INumber<T>
2  {
3          T Add(T arg1, T arg2);
4          T Mul(T arg1, T arg2);
5          T Div(T arg1, T arg2);
6          bool Et(T arg1, T arg2);
7          bool Lt(T arg1, T arg2);
8          T Abs(T arg);
9          T Neg(T arg);
10         T Zero { get; }
11         T One { get; }
12 };
```

Рис. 2. Пример кода обобщённой концепции универсального числового типа

Для комплексных чисел $z_1 = a + ib$ и $z_2 = c + id$ программная реализация операции деления осложняется тем, что правильность её результата зависит от значений знаменателей в слагаемых $\frac{ac+bd}{c^2+d^2}$ и $i\frac{bc-ad}{c^2+d^2}$. Согласно [3] выполнение может завершиться с ошибкой или привести к неверному результату, связанному с изменением точности, при достаточно большой или малой величине вещественной или мнимой части. Поэтому авторами использован улучшенный и отказоустойчивый алгоритм [4], который никогда не завершается неудачей и выдает результат с приемлемой точностью (Рис. 2).

Для реализации алгоритма с точки зрения обобщённого подхода от типов, представляющих мнимую или вещественную часть, требуется поддержка: $a + b$, $a - b$, $a * b$, a/b, $a \neq b$, $a \leq b$, $-a$ и $|a|$. Также известно, что, операция вычитания может быть выполнена с помощью операции сложения и аддитивной инверсии и не является дорогостоящей с точки зрения ее выражения в терминах уже существующих. А вот корректность операции деления, реализованной с использованием умножения и мультипликативной инверсии, может быть поставлена под сомнение. Для различных типов данных результат выполнения, казалось бы, подобных действий a/b и $a * (1/b)$ неодинаков и, например, для целых чисел всегда будет иметь ошибку округления при $b > 1$.

```
1  algorithm Z Div(Z₁, Z₂)
2        a = Z₁.Re    b = Z₁.Im
3        c = Z₂.Re    d = Z₂.Im
4        if (abs(d) <= abs(c))
5              Z = DivExt(a, b, c, d)
6        else
7              Z = DivExt(b, a, d, c)
8              Z.Im = -Z₃.Im;
9        end
10 end
11 sub Z DivExt(a, b, c, d)
12       r = d/c
13       t = 1/(c + d * r)
14       Z.Re = DivExt(a, b, c ,d ,r, t)
15       a = -a
16       Z.Im = DivExt(b, a, c ,d ,r, t)
17 end
18 sub x DivExt(a, b, c, d, r, t)
19       if (r != 0)
20             br = b * r
21             if (br != 0)
22                   x = (a + br) * t
23             else
24                   x = a * t + (b * t) * r
25             end
26       else
27             x = (a + d * (b / c)) * t
28       end
29 end
```

Рис. 3. Псевдокод безошибочного и отказоустойчивого алгоритма

Как видно из примера кода (Рис. 1) обобщённая концепция `INumber<T>` содержит в своем описании несколько операций и свойств, способных, в определенных сочетаниях, производить специализированные вычисления. Из неё сознательно исключены операции вычитания и мультипликативной инверсии, так как первая, как уже говорилось, легко может быть заменена сложением с отрицанием для второго операнда `Add(arg1, Neg(arg2))`, а вторая – делением нейтрального элемента мультипликативной группы на значение аргумента `Div(One, arg)`.

Конечное множество арифметических действий, обеспечивающие правильность работы алгоритма деления комплексных чисел (Рис. 2), выражено в терминах обобщённой концепции, которая должна поддерживать:

1. Операции сложения и умножения (строки 3-4), а также их коммутативность, ассоциативность и дистрибутивность.
2. Операцию деления (строка 5).
3. Операции сравнения по критерию «эквивалентность» (строка 6) и критерию «меньше чем» (строка 7).
4. Унарные операции получения абсолютного значения (строка 8) и отрицания (строка 9).
5. Двух нейтральных элементов аддитивной и мультипликативной групп (строки 10-11).

Понятие концепции в том или ином виде используется во многих языках программирования и библиотеках, но лишь немногие предлагают средства для их явного выражения. Базовая обобщённая реализация класса комплексного числа на языке C# использует концепцию универсального числового типа для экстернализации операций, где присутствуют обращения к параметрам типов, с последующим делегированием исполнения экземпляру концепции, соответствующей конкретному типу, и отвечающей предъявляемым семантическим и синтаксическим требованиям (Рис. 3).

```
1  abstract class ComplexBase<C, T> : IComplex<T>
2        where C : IComplex<T>
3  {
4        /* other methods and properties */
5        protected virtual C Div(C other)
6        {
7              if (!_cpt.Lt(_cpt.Abs(other.Re), _cpt.Abs(other.Im)))
8                    return DivExt(Re, Im, other.Re, other.Im, false);
9              return DivExt(Im, Re, other.Im, other.Re, true);
10       }
11       protected virtual C DivExt(T a, T b, T c, T d, bool neg)
12       {
13             var r = _cpt.Div(d, c);
14             var t = _cpt.Div(_cpt.One, _cpt.Add(c, _cpt.Mul(d, r)));
15             var re = DivExt(a, b, c, d, r, t);
16             var im = DivExt(b, _cpt.Neg(a), c, d, r, t);
17             return Create(re, neg ? _cpt.Neg(im) : im);
18       }
19       protected virtual T DivExt(T a, T b, T c, T d, T r, T t)
20       {
21             if (!_cpt.Et(r, _cpt.Zero))
22             {
23                   var br = _cpt.Mul(b, r);
24                   if (!_cpt.Et(br, _cpt.Zero))
25                         return _cpt.Mul(_cpt.Add(a, br), t);
26                   return _cpt.Add(_cpt.Mul(a, t), _cpt.Mul(_cpt.Mul(b, t), r));
27             }
28             return _cpt.Mul(_cpt.Add(a, _cpt.Mul(d, _cpt.Div(b, c))), t);
29       }
30 };
```

Рис. 4. Фрагмент кода базовой абстракции комплексного числа

В рамках работы были проведены эксперименты и выполнены проверочные тесты [5] по делению комплексных чисел z_1 и z_2 для предлагаемого подхода, а также нескольких специализированных программных платформ и библиотек. Как видно из результатов (Таблица 1) наибольший диапазон возможных значений для различных типов данных предоставляет обобщённая реализация комплексного числа, использующая концепцию универсального числового типа.

Существуют и другие реализации, применяющие подобный подход для конструирования обобщённых абстракций на языке программирования не в полной мере поддерживающем обобщённую парадигму [6].

Таблица 4. Результаты тестов деления комплексных чисел

Реализация	Операция деления				Тип данных
	Вещественная часть		Мнимая часть		
	Re_1	Re_2	Im_1	Im_2	
	14	36	57	82	
Mathcad	0.6456359102244389		0.11271820448877808		**double**
Matlab	0.645635910224438902743142144638 4		0.112718204488778054862842892768 08		**double**
NET	0.645635910224439		0.112718204488778		**double**
MathNET	0.6456359		0.1127182		**float**
PIT2019	0.6456359		0.112718195		**float**
	0.64563591022443889		0.11271820448877803		**double**
	0.64563591022443890274314214 53		0.11271820448877805486284289 29		**decimal**
	0.64563591022443890274314214 46384039		0.11271820448877805486284289 27680798		
	90024937655860349127182044887780548		00498753117206982543640897755610972		**BigDecimal**
	62842892768079800498753117206982543		56857855361596009975062344139650872		
	64089775561097256857855 4		81795511221945137157 1072		

Многим специалистам может показаться, что предлагаемая в результатах тестов точность может никогда не потребоваться и что диапазона значений встроенного типа **double** вполне достаточно для решения большинства вычислительных задач. В некоторых ситуациях они, вероятно, и окажутся правы, но осознанное упущение потенциальной возможности никак не должно оправдывать последующую неспособность программной инфраструктуры произвести требуемые вычисления при возникновении необходимости.

Литература

1. Vandevoorde D. C++ Templates: The Complete Guide / D. Vandevoorde, N.M. Josuttis, D. Gregor. – Boston: Addison–Wesley, 2017. – 832 p.

2. Stepanov, A. From mathematics to generic programming / A. Stepanov, E. Rose. – Boston: Addison–Wesley, 2015. – 293 p.

3. Baudin M. Error bounds of complex arithmetic / M. Baudin // NISP Toolbox Manual, 2011. – P.1-30.

4. Baudin M. A Robust Complex Division in Scilab / M. Baudin, R.Smith // CoRR. – 2012 abs/1210.4539.

5. URL: https://bitbucket.org/dyablokov-SSAU/pit2019/src

6. URL: https://archive.codeplex.com/?p=bignumber