УДК 004.4’242

ГЕНЕРАЦИЯ ИСХОДНОГО КОДА БАЗОВЫХ АЛГОРИТМОВ НА ОСНОВЕ ВЫСОКОУРОВНЕВЫХ СПЕЦИФИКАЦИЙ

Кустов М.Н., Дацун Н.Н.
Донецкий национальный технический университет, Украина

Обоснован выбор SADT-методологии как основы создания высокоуровневых спецификаций. Расширен набор базовых алгоритмов, для которых выделены инварианты по отношению к структурам данных. Показана возможность создания более сложных алгоритмов на основе базовых. Предложен алгоритм генерации исходного С-кода по спецификации SADT.

Введение

Разработка программного обеспечения предполагает участие огромного числа инженеров различных направлений, которые вручную создают программный продукт. В связи с этим, в процессе разработки программных систем возникает множество проблем: несогласованность структурных частей, двусмысленность, избыточность (неполнота) сопровождаемой проектной документации и др. Это связано с большим размером и высокой сложностью программного обеспечения [1].

Одним из самых трудоёмких и дорогостоящих этапов разработки программного продукта является создание функционала (написание программного кода). Именно поэтому областью исследования была выбрана автоматическая генерация программного кода [1].

Существует множество способов автоматической генерации программного кода. Каждый способ использует определённый набор входных данных, на основе которых и создаются инструкции на определённом языке программирования. Среди различных техник автоматической генерации исходного кода подробнее рассмотрим предметно-ориентированное моделирование (Domain-specific modeling (DSM)) и CASE-средства.

DSM представляет собой методологию проектирования и разработки программного обеспечения. Включает в себя использование предметно-ориентированного языка (domain-specific language (DSL)) для представления различных аспектов системы. DSM работают на более высоком уровне абстракции, чем моделирующие языки общего назначения, поэтому требуют меньше усилий и низкоуровневых деталей для описания системы. Указание решения непосредственно с помощью концепций предметной области позволяет повысить уровень абстракции вне программирования средствами DSM. По этим высокоуровневым спецификациям генерируется конечный продукт [2]. Модели служат не только как механизм для более простого понимания предметной области и её описания, но и как исходные данные для генерации полного, рабочего исходного кода. Такая автоматизация направлена на повышение продуктивности, качества и скрытия сложности системы.

Одним из широко известных CASE-средств является Rational Rose [3]. Этот продукт предназначен для автоматизации процессов анализа и проектирования ПО, но

также является мощным инструментом для автоматической генерации программного кода. Исходные данные представляются в виде UML-диаграмм, которые являются хорошим средством для моделирования различного рода объектов, процессов и т.п.

Проанализировав вышеописанные технологии, можно выделить наиболее полезные достоинства этих систем для дальнейшей генерации программного кода, а именно: использование графических нотаций и повышение уровня абстракции. На основе этих критериев была выбрана методология спецификаций SADT.

Методология SADT

Методология SADT может быть использована как инструмент генерации программного кода. Сочетание формальности обозначений и единообразного формализма для представления задач различных классов позволяет точнее формулировать спецификацию каждой из решаемых подзадач. Связность блоков гарантирует точность реализации взаимосвязей между вспомогательными алгоритмами в большой задаче. Разделение данных и функциональности позволяет проводить исследования полученного решения задачи на различных структурах данных. Далее рассмотрим практическое применение SADT-методологии для генерации программного кода [4].

Инварианты алгоритмов

В первую очередь следует определить инварианты алгоритмов, которые будут использованы в процессе генерации кода. Инвариантом алгоритма будем называть часть базового алгоритма, которая остаётся неизменной для различных структур данных [5].

Рассмотрим работу методологии на примере алгоритмов поиска и дальнейшего удаления элемента из одномерного массива, списка и строки.

Проанализировав простейшие алгоритмы для различных структур данных (рис. 1), мы видим, что некоторые части этих алгоритмов идентичны (на рис. 1 выделены серым фоном). Остальные же части специфичны для каждой отдельной структуры данных [5]. С помощью методологии SADT можно изобразить процесс обработки данных, используя базовые алгоритмы в качестве «строительного материала»(рис. 3).

```
/* структура данных -
одномерный массив*/

f = 0;
Res = -1;
i = 0;

while ((i < n) && !f)
{
r = x[i];
if (r == a)
{
Res = i;
f = 1;
}
else
i++;
}
```

```
/* структура данных –
односвязный список */

f = 0;
Res = -1;
i = p; // p – указатель на
голову списка
while ((i != NULL) && !f)
{
r = i->info
if (r == a)
{
Res = i;
f = 1;
}
else
i = i -> next;
}
```

```
/* структура данных –
строка */

f=0;
Res=-1;
i=0;

while (i<strlen(str) && !f)
{
r = str[i];
if (r == a)
{
Res = i;
f = 1;
}
else
i++;
}
```

Рисунок 1. Алгоритм поиска перебором для различных структур данных

i=0;
 	
while (i<Res)
i++;
while (i<n)
{
a[i]=a[i+1];
i++;
}
	i=p; // p – указатель на голову списка
	while (i!=Res)
	i=i->next;
	while (i!=NULL)
	{
	i->info=i->next->info;
	i=i->next;
	}
		i=0;
		
		while (i<Res)
		i++;
		while (i<strlen(str))
		{
		a[i] = a [i+1];
		i++;
		}

Рисунок 2. Алгоритм удаления элемента со сдвигом для различных структур данных

Для алгоритма удаления элемента инварианты можно выделить только для одномерного массива и строки, алгоритм для односвязного списка специфичен.

Следуя рассмотренному принципу, также можно выделить инварианты алгоритмов и построить SADT-диаграммы для других структур данных (файлов прямого и последовательного доступа, деревьев) [5].

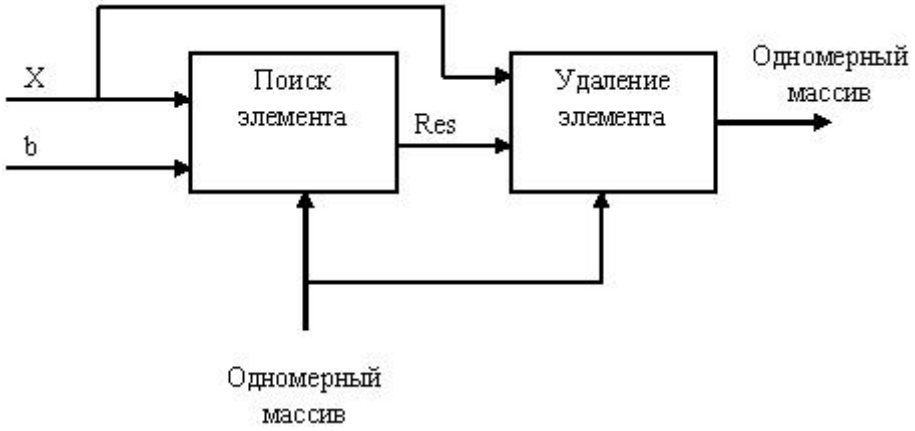

Рисунок 3. SADT-диаграмма поиска и удаления элемента в одномерном массиве

Совокупность простейших базовых алгоритмов для различных структур данных можно применять для решения более сложных реальных задач. Например, в методе сортировки «линейным выбором» применимы базовые алгоритмы «поиск минимального элемента» и «перестановка элементов местами». На рис. 4 изображена SADT-диаграмма формирования этого алгоритма. На рис. 5 показана декомпозиция алгоритма сортировки методом линейного выбора до простейших алгоритмов.

Рисунок 4. SADT-диаграмма сортировки элементов одномерного массива методом линейного выбора

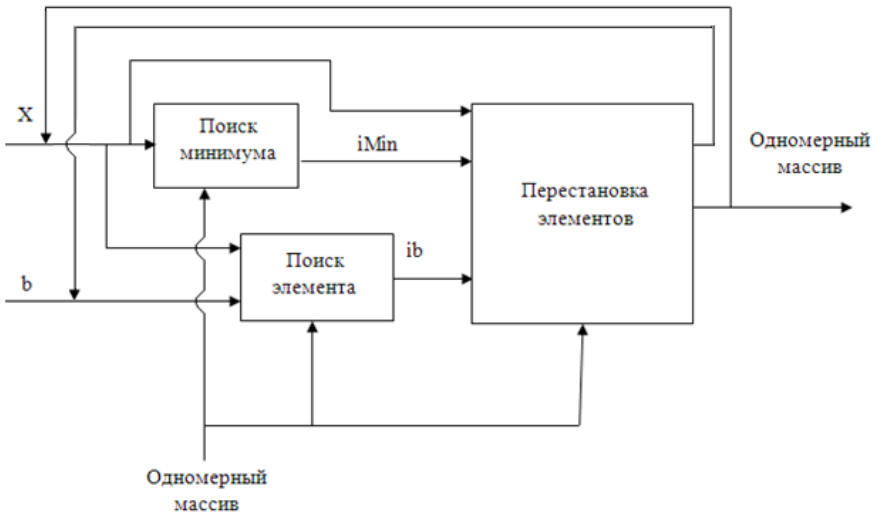

Рисунок 5. SADT-диаграмма декомпозиции алгоритма сортировки методом линейного выбора

Принцип декомпозиции является одним из основных в методологии структурного анализа и проектирования. Используя данный принцип, можно свести сложные алгоритмы к простейшим базовым алгоритмам. Например, задача первичной обработки экспериментальных данных сводится к простейшим алгоритмам «поиск максимального значения», «поиск минимального значения», «удаление элементов».

Программная реализация

Для реализации задачи автоматической генерации программного кода по SADT-методологии, был выбран язык C#.

В главном окне программы пользователь создает SADT-диаграмму, манипулируя элементами управления, выбрав соответствующий алгоритм, структуру данных, а также дополнительные условия. Алгоритм работы программы приведен на рис. 6.

В результате работы программы формируется файл GeneratedCode.cpp. Этот файл выступает в роли подключаемого модуля к любому проекту на языке C++. Подключив этот файл, его функционал можно использовать в любом месте программы как подключаемую функцию, передавая в неё соответствующие аргументы.

Алгоритм работы

1. Создание SADT-диаграммы;
   1.1 Выбор базового алгоритма из набора базовых алгоритмов;
   1.2 Выбор структуры данных, для которой будет сгенерирован программный код;
   1.3 Выбор дополнительных условий, которым должен соответствовать результат работы алгоритма
2. Генерация исходного кода;
   2.1 Выбор и чтение из файла-шаблона «скелета» в соответствии с выбранным алгоритмом;
   2.2 Чтение «начинки» алгоритма из файла-шаблона в соответствии с выбранной структурой данных;
3 Запись сгенерированного программного кода в файл.

Выводы

Расширен набор базовых алгоритмов, для которых выделены инварианты по отношению к структурам данных. На основе этих алгоритмов по методу композиции показана возможность создания более сложных алгоритмов. Предложен алгоритм генерации исходного кода по спецификации SADT. Результатом исследования является программный продукт, который демонстрирует принципы автоматической генерации исходного кода на языке С. Это ПО может быть использовано в образовательных целях.

Список источников

[1] Кустов М. Повышение эффективности алгоритмов генерации программного кода по методологии SADT. Электронный ресурс. Режим доступа http://masters.donntu.edu.ua/2012/fknt/kustov/diss/index.htm

[2] Kelly S., Tolvanen J.-P. Domain-Specific Modeling: Enabling Full Code Generation. - Wiley-IEEE Computer Society Press, 2008. – 448 p.

[3] Quatrani T. Visual Modeling with Rational Rose 2002 and UML, Addison-Wesley Professional, 2003. – 256 p.

[4] Кустов М.Н. Повышение эффективности алгоритмов генерации программного кода по методологии SADT. – Информационные интеллектуальные системы / Материалы XVI международного молодежного форума «Радиоэлектроника и молодежь в XXI веке». — Харьков: ХНУРЭ, 2012. – Том 6, С. 228–229.

[5] Kustov M., Guban B., Datsun N. Application of SADT for source code generation in learning the programming fundamentals/ Proceedings of the 6th Spring/ Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2012). – Perm, Russia. – p.28-33.

[6] Ross D. Structured Analysis (SA): A Language for Communicating Ideas// IEEE Transactions on Software Engineering. – 1977, V. SE-3, N. 1. – pp. 16-34.

[7] Marca D.A., McGowan C.L. IDEF0 and SADT: a modeler's guide. – Auburndale: OpenProcess, Inc., 2006. – 392 p.