МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ

Вестник Сыктывкарского университета.
Серия 1: Математика. Механика. Информатика.
Выпуск 4 (25). 2017

УДК 004.021, 004.043, 378

МЕТОДИЧЕСКИЕ ОСОБЕННОСТИ ПРИМЕНЕНИЯ СТРУКТУРНОГО ТИПА ДАННЫХ В ПРОГРАММАХ, НАПИСАННЫХ НА ЯЗЫКАХ СИ И СИ++

П. А. Макаров

В работе рассматриваются некоторые особенности методики преподавания языков программирования Си/Си++ студентам физико-математических специальностей вузов. Обсуждается применение структурного типа данных в программах как средство логической организации решения задачи. Описываются особенности перехода от процедурной парадигмы программирования к объектно-ориентированной.

Ключевые слова: процедурная и объектно-ориентированная парадигмы программирования, структурный тип данных, методы, конструкторы, перегрузка операций.

1. Введение

Программирование для студентов математических и физических специальностей — это одна из важнейших дисциплин, позволяющая глубже раскрыть содержание всех основных предметов образовательной программы. Параллельное изучение программирования оказывается полезным при освоении предметов как общего математического цикла, так и специальных курсов физико-математической и технической направленности.

В данной работе не рассматриваются механизмы защиты данных, классы, подробности создания конструкторов и деструкторов объектов и другие относительно сложные понятия, свойственные для объектно-ориентированной парадигмы (ООП) программирования. Все эти вопросы достойны более подробного и детального обсуждения, так как методика преподавания соответствующих тем студентам имеет множество особенностей [1].

По нашему мнению, удачным методическим решением при изучении языков программирования Си и Си++ является выбор в качестве основных учебных пособий классической книги Б. Кернигана и Д. Ритчи [2] и очень лаконичного введения в язык Си++ А. В. Столярова [3]. Конечно, одним только выбором учебных пособий методика преподавания предмета не ограничивается. Опыт практического решения разнообразных задач вместе с хорошей теоретической подготовкой играет основную роль при изучении программирования [4, 5]. При этом весьма важно то, чтобы решаемые студентами задачи были близки к их области специализации.

В данной работе основной акцент делается именно на методике обучения студентов использованию структур в конкретных задачах. С одной стороны, это позволяет повысить уровень абстракций при написании программ, что улучшает понимание студентами алгоритмов и логику работы программы. С другой стороны, применение структур — это прекрасный методический пример, показывающий студентам некоторую ограниченность процедурной парадигмы программирования (и, как следствие, языка Си), а также возможные пути решения указанной проблемы. Для этого, в частности, обсуждаются некоторые элементы ООП и средства их поддержки в языке Си++.

2. Высокий уровень абстракций и структуры

Практический опыт преподавания программирования показывает, что для решения подавляющего большинства математических, физических и технических задач удобно оперировать понятиями более высокого уровня абстракции, чем числа различных типов, символы или строки. В качестве некоторых примеров можно привести следующие объекты программ:

- геометрическая точка P в евклидовом (или псевдоевклидовом) пространстве R^n;
- элементарная частица массы m, обладающая электрическим зарядом q, спином s и временем жизни τ;
- IP-адрес устройства в компьютерной сети TCP/IP.

Программируя на языках Си и Си++, требуемого уровня абстракции можно достичь, используя структурный тип данных `struct`. На этом сходство данных языков программирования заканчивается и возникают достаточно существенные отличия.

В языке программирования Си ключевое слово `struct` вводит новый тег структуры, а не полноценный структурный тип. Поэтому введение нестандартных типов данных обычно оказывается удобным упростить с помощью оператора `typedef` [2].

Программирование структур на языке Си++ имеет свои отличительные особенности [1, 3]. В частности, с технической точки зрения в Си++ отпадает необходимость использовать ключевое слово `typedef`, так как идентификатор, стоящий после слова `struct`, непосредственно представляет собой имя нового типа. Более того, для поддержки программирования в стиле ООП, в языке Си++ структуры рассматриваются как совокупность свойств и методов. Методами, как известно [1,3], называются функции-члены структуры. Такой подход позволяет существенно повысить уровень абстракции понятий, используемых в программе.

3. Применение структур для решения задач

Рассмотрим конкретные методические особенности преподавания структур при изучении языков программирования Си и Си++ студентами физико-математических специальностей. В качестве примера рассмотрим одну из базовых задач в области геометрии.

Как известно из аналитической геометрии, прямую, проведённую через две заданные несовпадающие точки $M_1(x_1, y_1, z_1)$ и $M_2(x_2, y_2, z_2)$, можно определить следующей системой уравнений:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}. \tag{1}$$

Таким образом, множество точек с координатами (x, y, z), удовлетворяющими системе уравнений (1), образует прямую в пространстве R^3. На основе системы (1) можно сформулировать набор различных геометрических задач. В качестве конкретного примера рассмотрим следующую постановку задачи.

Задача 1. Даны координаты трёх точек A, B и C. Требуется определить, принадлежит ли точка C прямой AB.

Пример решения этой задачи на языке Си с использованием структур приведён в листинге 1. При разборе текста программы со студентами обязательно необходимо обсудить следующие ключевые моменты:

1. Применение оператора `typedef` в строках 2 и 5 при объявлении новых структурных типов данных `point` и `line`.

2. Использование введённых структурных типов в строках 11, 12 и 27.

3. Напоминание назначения строк 8 и 9, в которых описаны прототипы функций, используемых в главной функции программы.

4. Использование в строках программы 14, 16 и 19 функции `scanf()` не сопровождается проверками корректности ввода данных пользователем программы. Это может привести к двум типам ошибок при выполнении программы: чисто технической проблеме при неправильном наборе данных пользователем и логической ошибке, состоящей в том, что пользователь по невнимательности введёт одинаковые координаты точек A и B. Следует продемонстрировать конкретные примеры ввода таких данных и попросить объяснить студентов происходящее.

5. После обсуждения возможных проблем из предыдущего пункта необходимо предложить студентам придумать способы их устранения. Пример кода, устраняющий чисто техническую проблему, приведён в листинге 2. Необходимо объяснить студентам, что подобные фрагменты всегда следует использовать в реальных задачах вместо строк 14, 16 и 19, реализующих простейший ввод.

6. Фрагмент программы, устраняющий проблему, связанную с возможной идентичностью точек A и B, приведён в листинге 3. При обсуждении этого фрагмента очень полезно обсудить некоторую «избыточность» логического выражения в операторе `if` с точки зрения ООП. Таким образом можно подвести студентов к некоторым идеям полиморфизма (в частности, перегрузки символов стандартных операций, в том числе операции сравнения `==`).

7. Следует раскрыть смысл строки 17, в которой фактически происходит инициализация полей структуры `line`, и провести параллели с конструкторами объектов в языке Си++.

8. Алгоритм работы функций `set_line()` и `belong_point()`, описанных в строках 26–30 и 31–39 соответственно. Для функции `belong_point()` необходимо провести параллели с методами в языке Си++.

9. Также нужно обратить внимание на строки 33 и 34 и пояснить, как этот текст программы соотносится с системой уравнений (1).

10. Объясняя строки программы 35–38 и 20–23, следует напомнить студентам об отсутствии в языке Си специального логического типа данных.

Листинг 1. Пример решения задачи 1 на языке Си

```
#include <stdio.h>
typedef struct {
    double x, y, z;
} point;
typedef struct {
    point M1, M2;
} line;
line set_line(point M1, point M2);
int belong_point(line MM, point P);
int main(void) {
    point A, B, C;
    line AB;
    printf("Type the coordinates of the point A: ");
    scanf("%lg %lg %lg", &A.x, &A.y, &A.z);
    printf("Type the coordinates of the point B: ");
    scanf("%lg %lg %lg", &B.x, &B.y, &B.z);
    AB = set_line(A, B);
    printf("Type the coordinates of the point C: ");
    scanf("%lg %lg %lg", &C.x, &C.y, &C.z);
    if(belong_point(AB, C))
        printf("The point C belongs to the line AB.\n");
    else
        printf("The point C doesn't belong to AB.\n");
    return 0;
}
line set_line(point M1, point M2) {
    line X;
    X.M1 = M1; X.M2 = M2;
    return X;
}
int belong_point(line MM, point P) {
    double a, b;
    a = P.x*(MM.M2.y-MM.M1.y) + P.y*(MM.M1.x-MM.M2.x) +
       MM.M2.x*MM.M1.y - MM.M1.x*MM.M2.y;
    b = P.x*(MM.M2.z-MM.M1.z) + P.z*(MM.M1.x-MM.M2.x) +
       MM.M2.x*MM.M1.z - MM.M1.x*MM.M2.z;
    if (a == 0 && b == 0)
        return 1;
    else
        return 0;
}
```

Листинг 2. Проверка на соответствие данных ожидаемому формату

```
1 if(scanf("%lg %lg %lg", &A.x, &A.y, &A.z) != 3) {
2     printf("Wrong input! Program aborted.\n");
3     return 1;
4 }
```

Листинг 3. Проверка на совпадение точек A и B

```
1 if(A.x == B.x && A.y == B.y && A.z == B.z) {
2     printf("Points are identical! Program aborted.\n");
3     return 2;
4 }
```

Методически целесообразно после проведённого со студентами анализа листинга 1 привести решение той же задачи на языке Си++. Пример такого решения с использованием структур и применением всех обсужденных ранее особенностей ООП приведён в листинге 4.

Листинг 4. Пример решения задачи 1 на языке Си++

```
1 #include <stdio.h>
2 struct point {
3   double x, y, z;
4   bool operator==(point Q) {
5     if(x == Q.x && y == Q.y && z == Q.z)
6       return true;
7     else
8       return false;
9   }
10 };
11 struct line {
12   point M1, M2;
13   line(point A, point B) {
14     M1 = A;
15     M2 = B;
16   }
17   bool belong_point(point P) {
18     double a, b;
19     a = P.x*(M2.y-M1.y) + P.y*(M1.x-M2.x) + M2.x*M1.y -
         M1.x*M2.y;
20     b = P.x*(M2.z-M1.z) + P.z*(M1.x-M2.x) + M2.x*M1.z -
         M1.x*M2.z;
21     if (a == 0 && b == 0)
22       return true;
23     else
```

```
24        return false;
25    }
26 };
27 int main(void) {
28    point A, B, C;
29    printf("Type the coordinates of the point A: ");
30    if(scanf("%lg %lg %lg", &A.x, &A.y, &A.z) != 3) {
31      printf("Wrong input! Program aborted.\n");
32      return 1;
33    }
34    printf("Type the coordinates of the point B: ");
35    if(scanf("%lg %lg %lg", &B.x, &B.y, &B.z) != 3) {
36      printf("Wrong input! Program aborted.\n");
37      return 1;
38    }
39    if(A == B) {
40      printf("Points are identical! Program aborted.\n");
41      return 2;
42    }
43    line AB = line(A, B);
44    printf("Type the coordinates of the point C: ");
45    if(scanf("%lg %lg %lg", &C.x, &C.y, &C.z) != 3) {
46      printf("Error - wrong input! Program aborted.\n");
47      return 1;
48    }
49    if(AB.belong_point(C))
50      printf("The point C belongs to the line AB.\n");
51    else
52      printf("The point C doesn't belong to AB.\n");
53    return 0;
54 }
```

Несмотря на то что новое решение задачи, представленное листингом 4, на пятнадцать строк больше исходного решения, приведённого в листинге 1, эту разницу оценивать некорректно, так как в новом решении учтены все замечания, сделанные ранее. Таким образом, длина обоих решений примерно сопоставима, однако логическая осмысленность второго текста гораздо выше. Это связано с тем, что в тексте программы 4 фактически описаны только два новых структурных типа данных и логически связанные с ними методы. Главная функция программы полноценно использует все особенности, заложенные при конструировании типов `point` и `line`.

Очевидно, что решение задачи 1 можно было организовать совершенно иначе, например в полностью процедурной парадигме без использования структур или с применением классов, однако такие решения выходят за рамки обозначенной в статье темы.

4. Выводы

Таким образом, в работе описаны методические особенности преподавания языков Си и Си++ студентам физикам и математикам. Основное внимание уделено применению структурного типа данных в программах как средству логической организации решения задачи. Также обсуждены методические моменты, позволяющие на конкретных примерах показать студентам преимущество перехода от процедурной парадигмы программирования к объектно-ориентированной.

Список литературы

1. **Эккель Б.** Философия C++. Введение в стандартный C++. 2-е изд. СПб.: Питер, 2004. 572 с.

2. **Керниган Б., Ритчи Д.** Язык программирования С. 2-е изд., перераб. и доп. М.: Вильямс, 2015. 289 с.

3. **Столяров А. В.** Введение в язык Си++ : учеб. пос. 3-е изд. М.: МАКС Пресс, 2012. 128 с.

4. **Салимов Ф. В., Бухараев Н. Р.** Из опыта преподавания курса «Алгоритмы и структуры данных» в Казанском федеральном университете // *Казанский педагогический журнал. № 4 (99). 2013. С. 46–54.*

5. **Абрамян М. Э.** Применение электронного задачника при проведении практикума по динамическим структурам данных // *Компьютерные инструменты в образовании. № 3. 2013. С. 45–56.*

Summary

Makarov P. A. Methodical of the using struct type in C/C++ programs

Some features of the methodology of teaching C/C++ programming languages to students of physical and mathematical specialties of higher educational institutions are considered. The application of the structural

data type in programs as a means of logical organization of the solution of the problem is discussed. The features of the transition from procedural programming paradigm to object-oriented programming are described.

Keywords: procedural and object-oriented programming paradigms, structured data type, methods, constructors, operators overloading.

References

1. **Eckel B.** *Filosofija C++. Vvedenie v standartnyj C++* (Philosophy of C++. Introduction to C++), 2-e ed, SPb.: Piter, 2004, 572 p.

2. **Kernighan B., Ritchie D.** *Jazyk programmirovanija* (C programming language), 2-e ed., M.: Williams, 2015, 289 p.

3. **Stolyarov A. V.** *Vvedenie v jazyk Si++* (Introduction in C++ language), 3-e ed, M.: Max Press, 2012, 128 p.

4. **Salimov F. B., Bukharaev N. R.** Iz opyta prepodavanija kursa «Algoritmy i struktury dannyh» v Kazanskom federal'nom universitete (From the experience of teaching the course «Algorithms and Data Structures» at the Kazan Federal University), *Kazan Pedagogical Journal*, № 4 (99), 2013, pp. 46–54.

5. **Abrahamyan M. E.** Primenenie jelektronnogo zadachnika pri provedenii praktikuma po dinamicheskim strukturam dannyh (The use of an electronic task book in a workshop on dynamic data structures), *Computer tools in education*, № 3, 2013, pp. 45–56.

Для цитирования: Макаров П. А. Методические особенности применения структурного типа данных в программах, написанных на языках Си и Си++ // *Вестник Сыктывкарского университета. Сер. 1: Математика. Механика. Информатика. 2017. Вып. 4 (25). С. 50–58.*

For citation: Makarov P. A. Methodical of the using struct type in C/C++ programs, *Bulletin of Syktyvkar University, Series 1: Mathematics. Mechanics. Informatics*, 2017, №4 (25), pp. 50–58.

СГУ им. Питирима Сорокина *Поступила 11.12.2017*