Нов Български Университет

департамент „Информатика“

Резюме на монографичен труд

Програмиране на C++ – примери и решения, част първа

гл. ас. д-р Ласко Маринов Ласков

Съдържание

A Общо представяне на монографията

A.1 Актуалност

Компютърното програмиране е една от основните дисциплини, които се изучават в университетските програми по информатика, компютърни системи и технологии, приложна математика, и сродните им специалности. Популярността на този широк дял от информатиката е изключително голяма през последните десетилетия и е естествено продиктувана от търсенето на специалисти, както в индустрията, така и в академичните, и научно-приложните среди.

Въпреки популярността на тази дисциплина, създаването на университетски курсове, които успешно да въвеждат студентите в нея, не е проста и еднозначно решима задача. Причините за трудностите са редица фактори, сред които са: (i) разнородната подготовка на начинаещите в областта на точните науки в средното училище; (ii) създаването на връзка с другите дисциплини, които се изучават в учебната програма (особено с курсовете по математика); (iii) изборът на подходящи примери, които да илюстрират изложения материал, но и да са на съответното университетско ниво; (iv) изборът на подходящ компютърен език и платформа, с помощта на които да се извършва необходимата практическа част от обучението. Основна трудност представлява и изборът на структура на понятията, последователността, в която да бъдат представени те, или с други думи, изборът на програмна парадигма, с която да започне обучението.

За голямо съжаление, нуждата от специалисти в индустрията не винаги води до повишаване на качеството на обучение във висшите учебни заведения. Често външният натиск от компаниите и икономическата обстановка заставят студентите да прекратят преждевременно своето следване или да гонят краткосрочни цели, като овладяването на популярни за дадения момент технологии. Подобни обстоятелства водят до не достатъчно задълбочени знания в областта и умения, които бързо престават да бъдат актуални. От своя страна, университетските програми, стремейки се да се доближат до очакванията на индустрията, често пренебрегват дисциплини от така наречените крайни компетенции и дисциплини, по-близки до науката като изкуствен интелект, разпознаване на образи и др., за сметка на курсове, които звучат популярно, и атрактивно, но на практика не допринасят за развитието на обучаващите се като информатици. Възможно е тук читателят да се запита, каква е причината да се подготвят специалисти по високо-технологични области, ако в страната компаниите са занимават главно с разработката на посредстве-

ни проекти? Аз бих отговорил, че ние трябва да дадем шанс за появата на високо-технологични компании, чрез подготовката на квалифицирани кадри в университетите.

Ролята на академичната общност е да балансира крайностите, като основната цел на обучението би трябвало да бъде не успехът на един или няколко представители на индустрията, а личният успех на всеки индивидуален студент. В частност, обучението по програмиране трябва да гарантира нужната степен на задълбоченост на предадените знания и умения, което да създаде основата, гарантираща дългогодишна реализация, и универсалност на дипломанта. Освен това поради своя изключително приложен характер, програмирането може да помогне при разкриването на практическия аспект на редица дисциплини, често погрешно възприемани като чисто теоретични от начинаещия студент. Такива са почти всички математически курсове, които се изучават през първите една-две години от следването, като математически анализ, линейна алгебра, аналитична геометрия, дискретна математика, вероятности и статистика и други.

A.2 Цел и задачи

Целта на монографията е да въведе читателя в областта на компютърното програмиране, като:

- да изгради пълна система от понятия, запознавайки читателя първо с процедурната парадигма за програмиране и изграждайки обектно-ориентираната парадигма на нейна основа;
- да илюстрира представения материал с примери и ясни, и подробни решения;
- приведените примери да създадат връзка с останалите математически дисциплини, които се изучават в учебната програма;
- представените решения трябва да създават добри навици и стил на програмиране, като писане на сигурен, и добре форматиран код.

Основен акцент са решенията на задачите, които са представени в текста. Всяко решение има за цел подробно да изследва конкретния пример и да покаже изчерпателно последователността от задача-модел-компютърна реализация-валидация.

А.3 Структура на монографичния труд

Монографията се състои от предговор, девет глави, първата от които увод, три приложения и списък с цитираната литература. Текстът е написан на английски език.

Всяка глава съдържа представяне на дадената тема и завършва с параграф, наречен „Упражнения“, който се състои от примери, които са изцяло или частично решени. Запазвайки номерацията на главите, разгледаните теми са:

1. Увод.
2. Първи стъпки в програмирането на C++.
3. Фундаментални типове данни. Числа и аритметични операции.
4. Символи. Въведение в символните низове.
5. Условни конструкции. Релационни и булеви оператори.
6. Циклични изчислителни процеси: while, do и for цикли.
7. Функции.
8. Масиви и вектори.
9. Въведение в класовете.

Приложение A разглежда приоритета на изпълнение на операторите. Приложение B разглежда символите в ASCII таблицата. Приложение C е посветено на стандартните вход и изход с помощта на функциите scanf и printf.

Б Кратко изложение на съдържанието на монографията

В следващите параграфи съдържанието на текста е представено глава по глава. Показани са основните резултати, изложени в монографията.

Б.1 Глава 1. Увод

В първата глава се формулират целите на монографията и се представя нейното съдържание. Обосновава се изборът на процедурната програмна парадигма, от която се развива концепцията за обектно-ориентирано програмиране (ООП). Подчертано е, че директното изучаване на ООП крие риска от пропуски в знанията и уменията на читателя. От друга страна, започвайки с процедурно програмиране, читателят изгражда пълна йерархия от понятия, която му позволява да усвои и други подходи за реализация на компютърни програми.

Аргументиран е и изборът на програмния език C++ като подходящ за поставените цели, тъй като той е хибриден език, т.е. поддържа множество програмни парадигми, сред които и процедурната, и ООП. Подчертана е неговата популярност, както и връзката му с много от съвременните езици за програмиране като Java, C#, Lua и дуги.

Б.2 Глава 2. Първи стъпки в програмирането на C++

Тази глава представлява въведение в компютърното програмиране с програмния език C++. Вместо стандартните интегрирани среди за разработка (IDE[1]), е представена операционната система Linux заедно с колекцията от компилатори GCC[2]. Описан е процесът на компилация на проста C++ програма през Linux терминала. Известната програма "Hello, World" е обяснена ред по ред. Основен акцент пада върху разбирането на процеса на компилация и средата за програмиране, състояща се от операционна система, компилатор, текстов редактор и програма за откриване на програмни грешки (дебъгер).

Б.3 Глава 3. Фундаментални типове данни. Числа и аритметични операции

Представени са основните концепции за променлива и фундаментален тип данни, които се използват за представянето на цели числа и числа с плаваща запетая. Описани са основните елементи, съставящи компютърната програма: изрази и инструкции. Обяснен е операторът за присвояване, както и концепциите за lvalue, и rvalue. Направено е въведение в основните аритметични операции, както и в някои математически функции, дефинирани в стандартната библиотека cmath.

[1] От англ. Integrated Development Environment.
[2] GNU Compiler Collection.

Б.4 Глава 4. Символи. Въведение в символните низове

Преди да се пристъпи към изучаване на символните низове, добре е да се обясни символния тип в езика C++: char. Важен аспект е, че символният тип е целочислен тип и всеки символ се представя чрез целочислена стойност в ASCII таблицата.

Символните низове са основен тип данни, също както са и числените типове, и са необходими дори при най-началните примери при изучаването на компютърното програмиране. От друга страна, низовете имат сравнително сложна структура, която не може да бъде представена без концепцията за масиви, които се изучават по-късно. Освен това, в езика за програмиране C++ няма примитивен тип данни символен низ, а има две основни представяния: STL[3] класа string и така наречените C-низове.

Последните представляват масиви от символен тип, завършващи със специалния символ за край на низ и тук са споменати само, за да се обяснят константите от тип символен низ. Класът srting е сравнително лесен за използване, дори без наличието за концепцията за класове и е подходящ за повечето примери, изучавани от начинаещите програмисти.

Б.5 Глава 5. Условни конструкции. Релационни и булеви оператори

Едно от основните свойства на компютърните програми е да изпълняват различни действия при наличието на различни обстоятелства, били те някакви външни условия или резултати от вътрешни изчисления в самата програма. Тази глава представя механизмите на езика за пренасочване на програмния поток: конструкциите if, if/else, switch и оператора ?:. Обяснени са релационните оператори, като е показано приложението им при сравняването на числа и символни низове. Лексикографската наредба и сравняването на символни низове са често срещани елементи в задачите за програмиране.

Специално внимание е обърнато на логическите оператори и връзката с булевата алгебра, която по-подробно се изучава в курсовете по дискретна математика.

[3] От англ. Standard Template Library.

Б.6 Глава 6. Циклични изчислителни процеси: while, do и for цикли

Следващият основен механизъм за управление на програмния поток са цикличните изчислителни процеси. С тяхна помощ програмата изпълнява многократно повтарящи се действия в зависимост от стойността на дадено условие. Циклите са средство за изпълнението на сложни изчисления и са в основата на реализацията на много алгоритми.

Тази глава запознава читателя и с концепцията за псевдо-случайни числа – механизъм, който често се употребява при симулации, и имитирането на случайни процеси. Приведени са прости примери като например симулация на хвърляне на зар, както и по-интересни задачи като класическо приложение на метода Монте Карло.

Б.7 Глава 7. Функции

В тази глава функциите са представени като основен градивен елемент на компютърната програма, което малко или повече завършва представянето на процедурната парадигма за програмиране. Разгледани са всички по-важни свойства на функциите в езика C++, като подаването на параметри по стойност и по референция, връзката с глобалните, и локалните променливи, стойностите, връщани от функциите. Материалът е илюстриран с редица интересни примери, започвайки от често споменавания алгоритъм на Евклид за най-голям общ делител и стигайки до алгоритъма за шифроване ROT13, и теста за прости числа на Ферма.

Б.8 Глава 8. Масиви и вектори

Важно свойство на компютърните програми е възможността за съхраняване и обработка на голям брой стойности от един, и същ тип данни. Масивите са един от примитивните механизми за осъществяването на тази задача. В случая на езика C++ разбирането и използването на масивите не е напълно лесно, поради тяхната дуалност с указателите, затова и тяхното изучаване трябва да стане на няколко етапа. В тази глава са представени статичните масиви и е обяснено използването на индексите за достъп до елементите на масивите. Представен е и STL класът vector като динамична алтернатива на масивите, която е сравнително лесна за използване, особено при писането на сигурен код. Разгледана е пак темата за C-низовете, този път наблягайки на връзката им с масивите.

Представени са и многомерните масиви, които са използвани за илюстрация на класически примери от линейната алгебра като алгоритъма за

умножение на матрици и метода за решаване на система линейни уравнения чрез Гаусова елиминация.

Б.9 Глава 9. Въведение в класовете

Последната глава от монографията представлява въведение в принципите на обектно-ориентираното програмиране, развивайки ги от до сега представената процедурна парадигма за програмиране. Представени са началните знания, необходими на читателя, за да реализира собствени класове на езика C++. Представена е и първата основна концепция на обектно-ориентираното програмиране – капсулирането.

В главата е обяснена и разделната компилация, която позволява на програмиста да раздели кода на програмата на C++ на множество файлове, съдържащи декларацията на класовете и файлове, съдържащи тяхната реализация.

Литература

[1] H. Abelson and G. J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, USA, 2nd edition, 1996.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[3] E. W. Dijkstra. Go To statement considered harmful. *Communications of the ACM*, 11(3):147–148, 1968. Letter to the Editor.

[4] C. Horstmann. *C++ for Everyone 2E Binder Ready Version*. John Wiley & Sons, 2nd edition, 2011.

[5] C. Horstmann and T. Budd. *Big C++*. John Wiley & Sons, 2nd edition, 2008.

[6] D. Montgomery and R. George. *Applied Statistics and Probability for Engineers*. Wiley, 5th edition, 2010.

[7] S. Prata. *C++ Primer Plus (Developer's Library)*. Addison-Wesley Professional, 6th edition, 2011.

[8] K. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Science/Engineering/Math, 7th edition, 2011.

[9] V. Shtern. *Core C++ – A Software Engineering Approach*. Prentice Hall, 2nd edition, 2000.

[10] B. Stroustrup. *The C++ programming language*. Addison-Wesley, 2000.