Списание „Математика, компютърни науки и образование“ Том 6, кн. 1, 2023
Journal of Mathematics, Computer Science and Education Vol. 6, No. 1, 2023

DOI: 10.54664/NKTH1863

ДИДАКТИЧЕСКИ ПОДХОДИ ЗА РАЗВИВАНЕ НА УМЕНИЯ ПО ПРОГРАМИРАНЕ НА C++

Мария Христова

DIDACTIC APPROACHES FOR DEVELOPING C++ PROGRAMMING SKILLS

Mariya Hristova

Abstract: *Didactics is one of the most important aspects of teacher training. It is of particular importance for his/her professional development. The use of didactic models in teaching can significantly improve learning, and therefore their use is preferable. The purpose of this study is to evaluate the learning processes of C++ programming. Through a survey, an analysis was conducted and the results concerning the difficulties of studying it were described. Several methods are also suggested for easier assimilation of the learning content.*

Keywords: *IT; projects; informatics; programming; education; teaching; C++.*

ВЪВЕДЕНИЕ

За да разбере дигиталния свят, човек трябва да има някаква представа какво е програмиране. Програмирането стои зад всички цифрови решения, софтуери и системи, които използваме [6]. То представлява задаване на команди, които искаме компютърът да прочете, обработи и изпълни, а когато зададем поредица от такива команди, които се изписват на предварително избран от нас език за програмиране, вече имаме цяла компютърна програма.

Преди десетилетия програмирането е било част от обучението в прогимназиален етап, но с течение на времето, в следствие на бурното му развитие, започва да се изучава и в по-горните курсове. Научната литература изобилства от източници, посветени на преподаването на програмиране. Предложени са различни подходи, аргументирани с емпирични доказателства и подкрепени с примери.

Програмирането е неделима част от учебните планове на компютърните специалности. Това обуславя нуждата на факултетите от преподаватели с високо ниво на подготовка, прилагащи доказана методика, която развива уменията и води до изграждане на необходимите компетенции у студентите.

Всеки учебен предмет има своята специфика на преподаване. В този доклад се поддържа тезата, че основният проблем при изучаването на езици за програмиране не е усвояването на концепциите, а несправянето на студентите с интегрирането на отделните части в една работеща програма.

Програмирането изисква абстрактно и алгоритмично мислене, както и умения за решаване на проблеми [2]. Думите „програмиране“ и „кодиране“ често се използват като синоними. Кодирането всъщност представлява преобразуването на готов, подробно специфициран дизайн в програмен код. Това е внедрителската част от програмирането. Но поради придобитата популярност, терминът „кодиране“ може да се използва за обозначаване на по-широкото понятие „програмиране“.

Литературата изобилства от предложения за иновации в преподаването на програмиране. Някои от тях неправилно са подценявани. В статията [3] например е описан анекдотичен пример, който показва, че писането на коментари от експертни програмисти всъщност затруднява процеса на програмиране, което кара студентите да ги избягват. Това обаче не е добра практика, защото ролята на коментарите е да помагат за ориентирането в хилядите редове код.

Настоящото изследване има за цел да предложи идеи за дидактически модел за уводен курс по програмиране на C++. Стремежът е повишено ниво на възприемане на учебния материал. Направен е анализ на наличната литература и са проведени анкети със студенти, преминали курса.

ИЗЛОЖЕНИЕ

За да може да бъде разбрано и усвоено правилно, учебното съдържание трябва да бъде добре подготвено, правилно структурирано и актуално. Когато става въпрос за обучение по програмиране, необходимо е често да се следи развитието на технологиите и да се търсят начини за интегрирането на новостите в съществуващите курсове. От преподавателите се очаква да бъдат старателно подготвени и да изберат най-подходящия начин за презентиране на всяка тема.

1. Трудности в обучението на студентите

Много студенти изпитват затруднения в усвояването на основните концепции на обектно ориентираното програмиране (ООП) и затова университетите масово прилагат подхода objects-first, предполагащ уводна дисциплина по ООП. Не рядко, особено в България, за основа се използва хибридният език C++, поддържащ множество стилове на програмиране – процедурно, обектно ориентирано, функционално, генерично, паралелно.

Начинаещите програмисти трябва да усвоят различни набори от концепции паралелно и да ги прилагат в практическа среда едновременно, което тласка когнитивните процеси на мнозина към претоварване. Много опростено, когнитивните процеси включват прехвърляне на информация в мозъка и превръщането ѝ в знания чрез способността на мозъка да обработва информация [4].

В [5] са анализирани резултатите от проведено проучване на трудностите, свързани с усвояването на C++ чрез уеб базиран въпросник. Един от най-очевидните резултати е, че учениците са оценили, че имат по-малко трудности, отколкото се заключава от отговорите на учителите. Предполага се, че това е резултат от факта, че студентите вярват, че са разбрали проблема, но учителите виждат оставащите недостатъци в програмирането на курсовата работа и изпитите.

Според [9] изучаването на езика C++ от начинаещи винаги е съпътствано с проблеми, най-вече по отношение на управление на паметта, работата с масиви и *null-terminated* низове. По време на преподаване на упражнения по „Програмиране на C++“ се стига до същия извод. Наблюденията сочат още, че студентите се затрудняват и при работа с класове и обекти, правилна употреба на типовете данни, разбиране на принципите на ООП – полиморфизъм, капсулиране и наследяване и разбиране на работата на конструктори и деструктори.

Таблица 1: Отговори на студентите

ВЪПРОСИ	ОТГОВОРИ					
	№ 1	№ 2	№ 3	№ 4	№ 5	№ 6
Полезни ли са лекциите?	Да	Да	Да	Да	Да	Да
Достатъчни ли са като часове лекциите?	Да	Не	Не	Да	Не	Да
Полезни ли са упражненията ?	Да	Да	Да	Да	Да	Да
Достатъчни ли са като часове упражненията?	Не	Не	Не	Да	Не	Да
Можете ли да прилагате ООП?	Да	Не	Да	Да	Не	Да

Можете ли да обясните основните етапи при създаване и изпълнение на една програма**?**	Да	Не	Не	Не	Не	Да
Разбирате ли същността на „тип данни“?	Да	Да	Да	Не	Не	Да
Знаете ли кои са модификаторите за достъп в C++?	Да	Да	Да	Да	Не	Да
Затруднявате ли се при работа с конструктори и деструктори?	Не	Не	Не	Да	Да	Не
Знаете ли за какво служат get и set методите?	Да	Не	Да	Не	Не	Да

Във връзка с изследването на мнението и удовлетворението на студентите от обучението се проведе анонимно анкетно проучване сред повече от 20 студенти, като в Таблица 1 са представени отговорите само на няколко от тях, които са избрани на случаен принцип. Целта на анкетата беше да се определи кои са ключовите проблеми, свързани с обучението по програмиране и конкретно с конструкциите на програмния език C++.

От получените резултати може да се направи извод, че лекциите и упражненията са много полезни, но по-голяма част от студентите имат нужда от повече практическа работа, която да затвърди всичко научено и преподадено по време на лекциите.

Фигура 1: Резултати от анкетното проучване за полезността на лекциите и упражненията по C++.

На Фигура 1 са посочени резултатите от анкетното проучване, свързано с полезността на лекциите и упражненията по C++. Повечето от анкетираните са на мнение, че трябва да има повече упражнения, за да може да се усвои по-добре учебното съдържание. Останалите са доволни от броя на лекциите и упражненията и не смятат, че има нужда от тяхното увеличаване. На въпросите, свързани с конкретни проблеми, като например „Можете ли да обясните основните етапи при създаване и изпълнение на една програма?“, преобладава като отговор „Не“, което доказва, че определените часове за упражнения не са достатъчни и студентите имат нужда от повече практика.

В следващите точки са разгледани подробно всички констатирани проблеми.

1.1 Затруднение при работа с класове и обекти

При работата с класове една от констатираните основни грешки е писането на всички класове в един-единствен файл, което затруднява ориентацията в кода. Добрата практика изисква разделянето на интерфейса и имплементацията на всеки клас в отделни хедър и сорс файлове.

Друга, често допускана грешка, е използване на грешен модификатор за достъп до компонентите на класовете. Няма как да бъде зададен модификатор „private“ и да се очаква компонентът (метод или поле) да бъде използван извън конкретния проект. За такива цели се използва модификатор „public“. В много ситуации студентите забравят или не са запомнили, че модификаторът по подразбиране е „private“ и когато се опитат да достъпят някой от елементите на класа нямат това право и започват да се чудят и да търсят къде точно им е грешката.

```
class Student {
private:
    string name; //име, вид
    int age;     // години
}
```

1. 2 Затруднение при използване на типове данни

Прави впечатление, че студентите не могат да определят кога и какъв тип данни да използват в дадена задача. Те трябва да бъдат наясно какъв е обхвата на типовете данни, за да не се затрудняват в последствие.

1.3 Затруднение при разбиране на принципите на ООП в C++ - полиморфизъм, капсулиране и наследяване

Добрите софтуерни приложенията изискват да бъдат написани чрез прилагане на доказаните шаблони за дизайн и ООП принципите. Това гарантира тяхната надеждност, ефективност, защитеност и високо качество. В началото на обучението си повечето студенти изпитват затруднение при разбирането и прилагането на тези принципи, особено в тяхното взаимодействие и синергична работа. Затруднения има и при прилагането на всеки един от принципите поотделно. Изисква се практика и умения, чрез които принципите се реализират.

1.4 Затруднение при работа с конструктори и деструктори

В началото студентите се научават да остойностяват полета директно в главния метод, което не е най-добрия начин, след това достигат до извода, че класът трябва да е капсулиран, което се постига с модификатор за достъп *private* на полетата и чрез методи (гетъри и сетъри) вече с *public* достъп.

За да се избегне повторяемост на код и добавяне на get и set методи за всяко поле идват така наречените конструктори, които представляват специални методи за инициализация. Първоначално студентите се затрудняват да ги пишат, защото те са свикнали да правят отделни методи за всяко поле, а работата на конструкторите е да се задават стойности за всички полета.

Затруднения се констатират и с инстанцирането на класовете – създаване на нов обект на класа и избор на правилния конструктор, който да го инициализира.

Основната цел на деструкторите е да освобождават ресурсите, заделени за даден обект. Най-често допусканата грешка с тях е пропускането на извикване на деструктор за динамичен обект, когато той вече не е необходим. Получава се в ситуации, когато се работи с много обекти. Студенти, които са изучавали в средното училище програмни езици като Java и C# знаят, че има Garbage Collector, който се грижи да рециклира обектите, към които вече няма референции. В следствие на това студентите често пропускат да дефинират деструктор за клас, ползващ динамична памет.

2. Задачи за изпълнение, трудности и очаквани резултати

В проучването [7], което изследва успешните подходи на преподаване и трудностите, пред които са изправени 339 учители по компютърни науки, преподаващи програмиране на начално и средно ниво в Обединеното кралство, се посочва, че част от трудностите, изпитвани от учителите, се дължат на самите тях, а не на учениците.

Вътрешните трудности са свързани с липсата на самочувствие на учителите по тези теми и неадекватните им познания при определяне на подходящата методика. Външните трудности са свързани с недостатъчни образователни ресурси. В допълнение, като свързани с учениците трудности се посочват когнитивните им способности и уменията за решаване на проблеми. В тази връзка се посочва, че за учителите е полезно да получат по-задълбочено обучение, да подобрят своите педагогически умения и да създадат повече учебни ресурси.

2.1. Изисквания към преподавателите по програмиране

За да бъдат успешни, преподавателите трябва:

– да изградят у студентите умения да класифицират правилно задачите и да избират подходящите езикови средства и алгоритми;

– да следят развитието на езиците за програмиране и да интегрират в учебните курсове новите им възможности;

– да осигурят необходимите научни, теоретични и практически познания;

– да използват съвременни интегрирани програмни среди, съобразени с особеностите на обучаваните;

– да формират научен мироглед сред студентите[8].

2.2. Очаквани резултати

В уводния курс по програмиране студентите трябва да придобият познания в следните области на компетентност – информатика, ООП, графичен потребителски интерфейс и алгоритми и структури от данни. Те трябва да могат да обясняват основните етапи при създаване и изпълнение на една компютърна програма и да описват основни начини за създаване, изпълнение и тестване на програмен проект в интегрирана среда за разработка с използване на визуални графични средства. Освен това трябва да разбират същността на типовете данни, да ги разграничават и да могат да ги използват в определена задача.

3. Дидактически подходи за развиване на програмистки умения

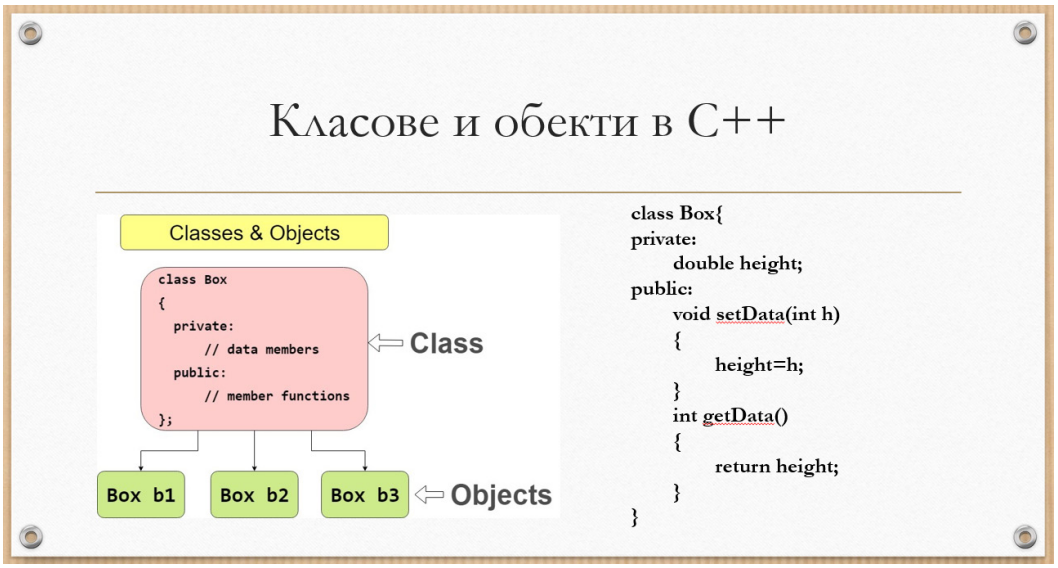
Двата основни фактора, от които зависи успешното усвояване на учебното съдържание във въвеждащите курсове по програмиране, са учебната програма и методите на преподаване [1]. Обучението по програмиране включва няколко дейности, като изучаване на езиковите характеристики, дизайн на програмата и разбиране на програмата.

Обичайният подход за преподаване на програмиране, залегнал в учебниците, е да се започне със синтаксиса на езика за програмиране и да се премине към свързаната с него семантика. Малко внимание се обръща на развитието на умения за програмно решаване на проблеми.

Има много различни начини, по които да се възбуди интереса на студентите или да им се помогне да разберат по-лесно учебното съдържание. Някои от тези начини са използване на презентации, работене по двойки, програмиране „на живо“ и използване на примери от реалния свят.

3.1 Използване на презентации

При предаване на нови знания презентацията дава възможност преподавателят да систематизира по по-ясен начин информацията и да я представи. Например класовете и обектите могат да бъдат по-добре представени чрез илюстрации както на Фигура 2. На фигурата е наблегнато на модификаторите за достъп *public* и *private*, които затрудняват студентите.



Фигура 2: Страница от презентация за класове и обекти

3.2 Работа по двойки

Това е практика, която е доста често срещана по време на упражненията – двама студенти споделят един компютър. Единият пише кода, а другият дава различни предложения и обратно. Тази практика е добра, защото студентите си помагат взаимно и успяват да изяснят погрешните си схващания един на друг, след като са стигнали до правилното решение на конкретната задача. Получените в този процес знания се оказват много трайни.

3.3 Програмиране „на живо“

За предпочитане е по време на упражненията да се решават поставените задачи стъпка по стъпка заедно със студентите. Това им дава възможност за съпреживяване на процеса на намиране на решение и незабавен отговор на възникнали въпроси по него.

3.4 Използване на примери от реалния свят

За обучението по обектно ориентираното програмиране е естествено използването на примери от реалния свят. Класовете и обектите представляват абстракции на реални обекти и процеси. Типични и подходящи са примерите с обектите хора, които имат характеристики като име, възраст, ЕГН, височина, тегло и др. Тези всички характеристики са подходящи за полета на клас – класът Човек, от който се инстанцират конкретни екземпляри (обекти), чрез които може да се представи някаква информация за всеки един от тях. Всеки един обект човек има конкретни стойности на описаните в класа характеристики. Тези стойности конституират състоянието на обекта.

Използването на такива примери помага и за по-лесното разбиране на това какво представлява полиморфизма. Полиморфизмът означава да има много форми. Може да се определи като техника, чрез която даден обект може да приеме много форми в зависимост от ситуацията [10]. Например една жена може да има много роли в различни ситуации и да показва различно поведение спрямо тях. За детето тя е майка, у дома – домакиня, в офиса – служител и т.н.

В педагогическата практика използването на примери от реалния живот е често прилаган метод на обучение, защото доказано помага за бързото усвояване на изучаваните концепции.

ЗАКЛЮЧЕНИЕ

По време на преподаване е много важно да се използват различни техники когато се вижда у студентите някакво неразбирателство относно учебното съдържание. Най-предпочитания подход от тях при упражненията е програмиране „на живо“, защото имат възможността да задават въпроси в момента на писане на дадена програма и да получат по-ясен отговор.

С помощта на анонимното анкетно проучване на студентите се достигна до отговора на въпроса „Какво най-много ги затруднява?“ и се определи кои са ключовите проблеми, свързани с обучението по програмиране на програмния език C++.

Факторите, които са причина за срещане на посочените трудности са много и различни. За да се намалят, преподавателите трябва да полагат непрестанно усилия да задържат вниманието на студентите и да им поднасят информацията систематизирано, използвайки различните дидактически подходи за развиване на програмистки умения, които са посочени в публикацията. Това не само ще им помогне да задържат тяхното внимание, но и да запомнят детайлите, които са им направили впечатление по време на упражненията или лекциите.

ЛИТЕРАТУРА

[1] **Ala-Mutka, K., Uimonen, T. & Jarvinen, H.M.** (2004). Supporting Students in C++ Programming Courses with Automatic Program Style Assessment.

[2] **Hiltunen T.**, “Learning and Teaching Programming Skills in Finnish Primary Schools – The Potential of Games”, University of Oulu, Master’s Thesis, 2016, http://jultika.oulu.fi/files/nbnfioulu-201605221873.pdf

[3] **Kaasbol J**., Exploring didactic models for programming, Department of Informatics, University of Oslo, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f5955ef9e44ea4b68da1ea315920021eeed5d55b

[4] **Malik S., Coldwell-Neilson Jo**, A model for teaching an introductory programming course using ADRI, 2016, https://www.researchgate.net/publication/295902185_A_model_for_teaching_an_introductory_programming_course_using_ADRI#pf1e

[5] **McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B., Laxer, C., Thomas, L., Utting, I. & Wilusz, T.** (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students, SIGCSE Bulletin, 33(4), pp. 125–180

[6] **Peter J. Rich, Scott Bartholomew, David Daniel, Kenzie Dinsmoor, Meagan Nielsen, Connor Reynolds, Meg Swanson, Ellyse Winward, Jessica Yauney**. (2022) Trends in tools used to teach computational thinking through elementary coding. *Journal of Research on Technology in Education* 0:0, pages 1-22, https://www.tandfonline.com/doi/full/10.1080/20004508.2019.1627844

[7] **Sentence, S. and Csizmadia**, **A.,** 2017. *Professional recognition matters: Certification for in-service computer science teachers* Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE

[8] **Б. К. Шаяхметова**. Особенности преподавания процедурно-ориентированных языков [Электронный ресурс] [2011 г.]. **B. K. Shayakhmetova**. Features of teaching procedurally oriented languages [Electronic resource] [2011]

[9] **Дончев И.,** Teaching C++ as a First Programming Language, Mathematics, Computer Science and Education, 2019, https://doi.org/10.54664/LxPYA6043 **Donchev I.,** Teaching C++ as a First Programming Language, Mathematics, Computer Science and Education, 2019, https://doi.org/10.54664/LxPYA6043

[10] Полиморфизъм в C ++, https://bg.myservername.com/polymorphism-c Polymorphism in C++, https://bg.myservername.com/polymorphism-c

ИНФОРМАЦИЯ ЗА АВТОРА

Ас. Мария Христова – докторант, специалност „Информатика и компютърни науки“, Факултет „Математика и информатика“, Великотърновски университет „Св. св. Кирил и Методий”, e-mail: maria_mt@abv.bg

ABOUT THE AUTHOR

Mariya Hristova – Lecturer, PhD student in Informatics and Computer Science, Faculty of Mathematics and Informatics, St. Cyril and St. Methodius University of Veliko Tarnovo, e-mail: maria_mt@abv.bg