АЛГОРИТМ ВИВЧЕННЯ С++ ЗА САМОВЧИТЕЛЕМ

Кобякова Л. М.

Державний заклад «Південноукраїнський національний педагогічний університет імені К. Д. Ушинського»

Як з'їсти слона?
Розрізати на шматки.
Послідовно та методично їсти по шматку.
Чарівне слово для досягнення успіху в будь-якій справі: ЩОДЕННО

У зв'язку зі скороченням кількості аудиторних годин (лекційних і практичних) актуально навчити і організувати студентів самостійно вивчати програмування.

Ми запропонували студентам і школярам алгоритм вивчення мови програмування С ++ за самовчителем Васильєва [1]. Наводимо його.

Підготовка:

1) Заведіть зошит для написання конспекту. Можна для кожного розділу заводити окремий 12-18-сторінковий зошит (на обкладинці кожного зошита напишіть Розділ №. Назва), або один 60-сторінковий зошит для Частини 1 (Процедурне програмування) і один 96-сторінковий – для Частини 2 (ООП).
2) Скачайте з офіційного сайту і встановіть на комп'ютер середовище програмування, наприклад, Qt Creator:

Як організований текст Самовчителя?

1) Текст Самовчителя складається з 3 частин:
   Частина 1 «Процедурне програмування в С++» (Розділи 1-6).
   Частина 2 «Об'єктно-орієнтоване програмування в С++» (Розділи 7-12).
   Частина 3 «Середовища розробки» (Розділи 13-14).
2) Розділи розбиті на:
   – параграфи (в розділі в середньому 4-13 параграфів);
   – приклади розв'язання задач (5-9 прикладів);
   – резюме;
   – контрольні питання;
   – задачі для самостійного виконання (20 в кожному розділі).
3) Параграфи (2-5 стор) містять теоретичний матеріал та лістинги.

Як опрацювати Розділ

Написане *курсивом* – це приклад виконаної згідно алгоритму дії.

1. Прочитайте назву Розділу і запишіть її в конспект.

Назва Розділу містить відповідь на питання: Що є головним (об'єктом розгляду) в цьому розділі.

Розділ 1. Основи мови С++

2. Перепишіть в конспект із Змісту список параграфів, з яких складається Розділ, та номера сторінок

Параграфи Розділу – це і є список *основ мови С++*

Залишайте після назви кожного параграфу декілька рядків.

Процедурне та об'єктно-орієнтоване програмування

...

Структура програми в С++

Створення простої програми

...

Мета наступних п.п.3-5 – «поставити в голові маячки на головне», на що звертати основну увагу, навколо чого крутиться викладання, виділити основні поняття, про які йде мова в розділі.

3. Прочитайте контрольні питання
4. Прочитайте резюме
5. Прочитайте весь розділ (бажано в один присід) і після прочитання кожного параграфа запишіть в залишені після його заголовку порожні рядки (тільки) назви основних понять

Процедурне та об'єктно-орієнтоване програмування

Процедурне програмування

Об'єктно-орієнтоване програмування

3 механізми ООП

У такий спосіб Ви виокремите головне, на що треба буде звернути увагу при подальшому вдумливому прочитанні і ретельному опрацюванні розділу

6. Скласти стислий конспект кожного параграфа. Як це робити?

Читаєте абзац, виділяєте і записуєте в конспект основну думку (бажано не переписувати, а сформулювати самостійно).

Читаєте 1 абзац Розділу 1:

Серед безлічі мов програмування С ++ займає особливе місце. Він досить простий, лаконічний і виключно ефективний. Мова С ++ створений професіоналами для професіоналів і є розширенням мови С для підтримки об'єктно-орієнтованої парадигми програмування.

Записуєте в конспект (наприклад):

С ++ - мова промислового програмування.

С ++ - розширення С для підтримки ООП.

7. В параграфах є приклади програм – лістинги. Їх потрібно опрацювати.
   1) Перед лістингом знаходиться умова задачі.
   2) Лістинг – програмний код.
   3) Після лістингу наведені пояснення технічних деталей коду

Як опрацювати лістинг?

1) Перепишіть умову задачі у вигляді:

Дано: вихідні дані

Знайти/Потрібно зробити: результат

Наприклад, Ви читаєте:

В наведеному коді за заданими сторонами трикутника обчислюється його площина.

Напишіть у вигляді:

Дано: сторони трикутника

Знайти: площину трикутника

2) Прочитайте пояснення к коду лістинга (після нього), при цьому відшукуйте у лістингу рядки, про які йдеться, продумуйте їх, тобто зіставляйте з відомим теоретичним матеріалом.

Як?

– Подумки відповідаючи на питання: це робиться так, тому що …

3) «Прочитайте» кожний рядок коду.

Як?

– Формулюйте про себе або пишіть на листку, що робить кожний рядок коду.

Наприклад:

1	#include <iostream>	*Підключення бібліотеки вводу-виводу*
2	#include <cmath>	*Підключення бібліотеки математичних функцій*
3	using namespace std;	*Інструкція компілятору використовувати стандартний простір імен*
4	main()	*Заголовок головної функції програми*
5	{	*Начало коду функції main()*
6	double a, b, c;	*Опис вихідних даних – дійсних змінних a, b, c (сторін трикутника)*
7	a = 3; b = 4; c = 5;	*Присвоювання значень змінним a, b, c*
8	double p = (a + b + c)/2;	*Опис дійсній змінної p та обчислення полупериметру трикутника за формулою* $p = \frac{a+b+c}{2}$
9	double s = sqrt(p*(p – a)*(p – b)*(p – c));	*Опис дійсної змінної s та обчислення площини трикутника за формулою Герона* $s = \sqrt{p(p-a)(p-b)(p-c)}$
10	cout << "s = " << s << endl;	*Вивід на екран площини трикутника у вигляді:* *s = значення змінної s*
11	}	*Кінець коду головної функції main()*

З одного боку, Ви бачите зразок грамотно написаного і оформленого коду. З іншого, - Ви вчитеся «читати» код, і це вміння виявиться Вам корисним, коли в коді потрібно буде знайти помилку. Прочитати код – це зрозуміти / прочитати не те, що Ви думаєте, що код робить, а те, що робить код насправді.

4) Після опрацювання лістинга, потрібно самостійно написати програму, яка розв'язує поставлену задачу. Для цього складаємо словесний алгоритм.

Як скласти словесний алгоритм?

– Словесний алгоритм складається із змістовних частин програми записаних зрозумілими для Вас словами і/або значками.

Аналіз:

Код приведеного Лістинга складається із наступних змістових частин:

Рядки 1-3 – підключення бібліотек <iostream>, <cmath>

Рядки 6-7 – опис та ініціалізація вихідних даних – сторін трикутника

Рядки 8-9 – обчислення полупериметру та площини трикутника

Рядки 10 – вивід на екран результату – площини трикутника

Словесний алгоритм можна записати, наприклад, так:

Словесний алгоритм	Пояснення до позначень:
1) <iostream> (cout, endl), <cmath> (sqrt)	Бібліотека <iostream> підключається для того, щоб використати cout, endl Бібліотека <cmath> тому, що формула Герону містить корінь
2) Δ: a, b, c	Вихідні дані: сторони трикутника a, b, c Зрозуміло, що a, b, c – дійсні числа. Якщо б це було б неочевидно, потрібно було б дописати a, b, c ∈ R. Ми не пишемо, що a = 3, b = 4, c = 5. Чому? У якості сторін трикутника можна взяти будь-які позитивні дійсні числа, що задовольняють нерівностям трикутника: сума будь-яких 2-х сторін більше третьої. Наприклад: 8, 2, 7
3)	Формулу записуємо в математичній формі
4)	
5) s → экран	s – це результат, його потрібно виводити на екран обов'язково

Закрийте Самовчитель та, дивлячись в Словесний алгоритм, напишіть програму самостійно на своєму максимумі. Що це означає?

Ви повинні розуміти, що завдання відтворення лістингу не ставиться, тому свою короткострокову пам'ять включати не треба, спрацює «правило 3-х зе»: «зазубрив-здав-забув». Ставиться завдання, щоб Ви написали код самостійно максимально добре, наскільки можете, і отримали досвід самостійного програмування.

Наприклад, так:

<iostream> (cout, endl), <cmath> (sqrt)	#include <iostream>
	using std :: cout;
	using std :: endl;
	#include <cmath>
	using std :: sqrt;
	main()
	{
Δ: a, b, c	// Опис та ініціалізація сторін трикутника
	double a = 3;
	double b = 4;
	double c = 5;
	// Обчислення полупериметра та площини трикутника за формулою Герона
	double p = (a + b + c)/2;

	double s = sqrt(p*(p – a)*(p – b)*(p – c));
s → экран	// Вивід на екран площини трикутника
	cout << "s = " << s << endl;
	}

Видно, що код, написаний за Словесним алгоритмом, відрізняється від коду лістингу.

- з бібліотек «взяті» тільки необхідні функції
- кожна з змінних a, b, c описана та ініціалізована окремо.
- код відкоментований.

Однак суть збережена (все зроблено).

8. **Приклади**. Ви опрацювали теоретичний матеріал і лістинги, наведені в параграфі. Наступне за планом - опрацювання прикладів. Приклади є практичними завданнями. Серед прикладів, як правило, є одне або декілька завдань, в яких акцент робиться на вивчення конкретного алгоритму (наприклад, алгоритм сортування масиву, рішення простого рівняння) і / або представлення даних і роботу з ними (наприклад, комплексні числа). Робота з прикладами проводиться так само, як і з лістингами.
9. **Резюме**. Читаєте. Якщо в цьому є необхідність, коригуєте свої записи в конспекті і уточнюєте свої знання.
10. **Контрольні питання**. Відповідаєте на них письмово на окремому аркуші (не в конспекті), не заглядаючи в Самовчитель і конспект. Після закінчення, шукаєте в Самовчителі відповіді на контрольні питання і порівнюєте з Вашими відповідями. Якщо відповідь неправильна або неповний, опрацьовує відповідний матеріал ще раз.
11. **Завдання для самостійного вивчення**. Розв'язуєте завдання, відібрані викладачем. Зрозуміло, що краще все.

Ми пропонували студентам і школярам старших класів слідувати запропонованому алгоритму. У тих, хто обрав навчатися за запропонованим алгоритмом, сформовані міцні теоретичні знання і стійкий навик осмисленого програмування.

Література

1. Васильев А.Н. Самоучитель C++ с задачами и примерами. – СПб: Наука и Техника, 2010. – 480 с.
2. Спрол Антон. Думай как программист: Креативный подход к созданию кода. C++ версия. – М.: Эксмо, 2018. – 272 с.
3. Оакли Барбара. Думай как математик. Как решать любые задачи быстрее и эффективнее. – М.: ООО «Альпина Паблишер», 2015.