С.А. Вернигоренко
НПУ імені М.П. Драгоманова

Вивчення основ об'єктно орієнтованого програмування у класах фізико-математичного профілю

Питання стосовно методики навчання основ програмування у загальноосвітній школі будуть завжди актуальними. Це пов'язано перш за все з тим, що наука інформатика взагалі і теорія створення програмних засобів зокрема стрімко розвивається. З'являються не лише нові середовища програмування, але й нові мови і технології програмування. Так серед професійних програмістів найбільш поширеною є мова Сі та її об'єктне розширення – мова Сі++. На сьогоднішній день існують Сі-подібні мови програмування такі, як Java, JavaScript, ActionScript, C#, C.net. З'явилися нові технології у програмуванні такі, як COM, DCOM, COM+, використання котрих надає можливість створювати програмні засоби із реалізацією ефективного обміну даними між ними. Наприклад, можливість використання таблиці, розробленої у програмі Excel, як вбудованого об'єкта у програму Word – це, фактично, застосування COM-технології у програмуванні. Існують способи створення елементів управління користувача, які базуються на застосуванні технологій ActiveX або ATL, котрі також є реалізацією COM-технології.

З огляду на вище вказане виникає думка про те, що окремих учнів варто знайомити з певними основами використання сучасних інформаційних технологій у програмуванні, оскільки це в подальшому має позитивно вплинути на їхній світогляд, вибір майбутньої професії, становлення їх, як знавців своєї справи. Поряд з цим слід зауважити, що отримання таких, дещо специфічних знань може бути цікавим і доступним не для всіх. Такий матеріал, перш за все, має бути орієнтований на учнів фізико-математичного та інформатичного профілю.

Як відомо, загальноприйнятою концепцією навчання інформатики у середній школі є підготовка користувача сучасної обчислювальної техніки. Але після закінчення школи деякі з цих користувачів прагнуть отримати вищу освіту з кваліфікацією "програміст". Тому загальноосвітня школа має підготувати таку категорію учнів до вивчення нових дисциплін, пов'язаних з програмуванням. На жаль, ситуація стосовно підготовки кваліфікованих програмістів у переважній більшості випадків складається наступним чином: у певних класах є один-два учні, що цікавляться програмуванням. В подальшому деякі з них стають професійними програмістами. На запитання, про те, хто навчив їх програмування, досить часто доводиться чути відповідь: "Школа мені практично нічого не дала у плані мого професійного становлення. Увесь матеріал я опановував самотужки." Після навчання у ВУЗі часто доводиться чути аналогічні твердження. Саме з цих причин однією з основних задач вчителя має бути потреба у вивченні нових технологій та пошук нових методик навчання. Результатом такої роботи повинно стати збільшення кількості підготовлених учнів, виховання у них прагнення до опанування новими знаннями та вміння застосовувати їх при розв'язуванні поставлених задач.

Безперечно, існуючі методики навчання основ структурного та модульно-процедурного програмування дають позитивний ефект у процесі навчання. Тому для класів загального або гуманітарного профілю питання стосовно пошуку їм альтернативи не є першочерговим. Тут варіативною частиною може бути, зокрема, підбір вправ та задач для вироблення умінь та навичок написання алгоритмів певною мовою програмування. У цьому контексті вчителю може допомогти значна кількість підручників, навчальних посібників та лабораторних практикумів, присвячених цій тематиці.

Але учням, що навчаються у класах з поглибленим вивченням фізики, математики та інформатики, можна пропонувати нові шляхи вивчення основ програмування. Першим кроком при цьому може бути створення методики вивчення саме об'єктно-орієнтованого програмування, як основи усіх сучасних технологій програмування. Тут варто дати відповіді на три першочергові запитання: який матеріал обрати для вивчення, у якій послідовності здійснювати його подання, які вправи та задачі пропонувати учням для вироблення навичок написання програм.

Стосовно вибору мови програмування, з використанням якої здійснюватиметься навчання, можна наводити різні міркування "за" і "проти". Основними аргументами на користь вивчення мови Сі++ є такі. У цій мові найбільш повно реалізований об'єктно-орієнтований підхід. Існує значна кількість Сі-подібних мов програмування, тому, маючи знання з даної мови, можна значно швидше вивчити інші. Мова Сі++ за рахунок гнучкості та швидкості виконання програм, створених з її використанням, є мовою професійних програмістів.

Досить непростим є питання добору тем для вивчення учнями та порядок їх вивчення. Деякі вчителі пропонують спочатку ознайомити учнів із структурним та модульно-процедурним підходами у програмуванні (вивчити мову Сі) та лише після цього переходити до вивчення об'єктно-орієнтованого підходу (мова Сі++). Ця точка зору цілком виправдана, вона, зокрема, відображає історичний хід розвитку програмування. Але є певні зауваження стосовно такого підходу. Перш за все це кількість учбового часу. Вивчення лише одного структурного підходу потребує не менше півріччя з розрахунку від двох до чотирьох годин на тиждень. На ознайомлення з основами об'єктно-орієнтованого підходу потрібно витрати не менше двох-трьох місяців навчального часу. Інша причина – формування у учнів об'єктного підходу у програмуванні. Вважається, що такий підхід варто використовувати лише при написанні досить складних програм з великою кількістю різних структурованих типів даних. Поряд з цим необхідно пам'ятати про те, що об'єктне програмування, порівняно із структурним, краще дисциплінує програміста, зокрема, за рахунок чіткого розподілу даних та функцій їх опрацювання. Тому володіння навичками створення програм з використанням об'єктного підходу в подальшому надає можливість програмісту-практику більш ефективно організовувати дані у програмі та використовувати раніше створені структури даних при розробці нових проектів. З цих причин у нижче описаній методиці проведення уроків застосовується комбінований підхід: повідомляється як матеріал, що стосується структурного програмування, так і відомості про можливості застосування об'єктного підходу.

Важливим питанням є добір навчальних вправ і задач. Тут доцільно використовувати задачі, що розв'язуються у курсі структурного програмування, але розглядати їх через призму об'єктно-орієнтованого підходу. Поряд з цим корисно пропонувати учням задачі, у яких моделюються певні об'єкти чи процеси реального життя. На прикладі таких задач можна продемонструвати основні концепції та прийоми опрацювання даних засобами об'єктного програмування.

Нижче наведено орієнтовний перелік тем та приблизна кількість академічних годин, що відводяться на їх вивчення.

1. Основні уявлення про мови програмування 1 год.
2. Етапи розв'язування задач на ЕОМ. Алгоритм 2 год.
3. Структура Сі-програми. Препроцесор. Поняття функції 2 год.
4. Поняття змінної. Типи змінних мови Сі. Область дії змінної 1 год.
5. Поняття вказівника та посилання. Причини застосування вказівників 2 год.
6. Бібліотечні функції введення-виведення 2 год.
7. Поняття оператора. Лінійні оператори. Основні арифметичні операції та математичні функції 2 год.
8. Створення та застосування функцій користувача 2 год.
9. Тип даних структура 2 год.
10. Поняття класу та об'єкта 2 год.
11. Поняття умови. Основні логічні операції. Закони Де Моргана 1 год.
12. Оператори розгалуження та вибору 4 год.
13. Оператори повторення 6 год.
14. Тип даних масив. Зв'язок між вказівником та масивом 4 год.
15. Робота з рядковими величинами 2 год.
16. Робота з файлами 2 год.
17. Поняття конструктора та деструктора класу. Конструктор копіювання та конструктор перетворення 2 год.
18. Успадкування 2 год.
19. Перевизначені функції класу. Віртуальні функції 4 год.
20. "Дружні" функції та "дружні" класи. Попереднє оголошення заголовків класів 1 год.
21. Перевантаження стандартних операцій 4 год.
22. Створення та використання шаблонів функцій та класів 4 год.

Всього 54 години навчального часу. Сюди варто додати чотири години на виконання залікових самостійних робіт – матимемо 58 годин. Якщо за державною програмою відводиться дві години на тиждень обов'язкового часу на вивчення інформатики та використати дві години додатково, як заняття на факультативі, то з розрахунку сімнадцяти навчальних тижнів на півріччя отримуємо 68 годин. Тобто, відповідно до пропонованого плану навчання, залишається десять годин резервного часу, який можна використати на розв'язування задач.

Далі коротко розкрито суть навчального матеріалу, що виноситься для засвоєння учнями та вироблення умінь і навичок створення програм.

Варто повідомити учнів про основні поняття, що використовуються у процесі навчання програмування. До таких понять, зокрема, відносяться "програма", "мова програмування", "алфавіт мови", "синтаксис мови", "семантика мови", "середовище розробки", "інтерпретація", "компіляція", "трансляція". Можна також познайомити учнів з деякими класифікаціями мов програмування (за структурою, типізацією і т. д.) [4, 41].

Після цього необхідно ознайомити учнів з етапами розв'язування задач на програмування, охарактеризувати їх, навести певні аналогії розв'язування задач з використанням цих етапів. Далі потрібно пояснити на прикладах поняття алгоритму та його властивостей.

Наступним кроком має бути ознайомлення учнів із структурою програми, описаної мовою Сі. Тут потрібно розповісти про розділи, з яких складається опис програми, та можливості розташування цих розділів при її описуванні. Учні мають вивчити, що таке препроцесор та вміти використовувати його основні директиви. Далі можна повідомити учням суть понять "підпрограма" (функція), головна програма, навчити записувати заголовок і тіло головної програми.

На наступному уроці можна розпочати вивчення понять змінної та константи, основних атрибутів змінної (ім'я, тип, значення, адреса). Учні повинні навчитися використовувати у програмі змінні простих типів (*char, int, float, double*), розуміти, що таке область дії змінної.

Для вироблення практичних навичок створення найпростіших програм можна запропонувати учням вправи на різні способи описування та ініціалізації змінних і констант. Особливу увагу потрібно звернути на ознайомлення учнів з можливостями покрокового виконання програми та перегляду значень змінних, використовуючи засоби середовища розробки. Якщо учні опанують подібні засоби, то можна очікувати, що вони краще розумітимуть структуру програми та суть вказівок, які застосовуються у ній.

Оскільки у мовах Сі та Сі++ часто доводиться оперувати таким поняттям, як вказівник, то бажано пояснити учням його смисл, починаючи з перших уроків, тобто одразу після повідомлення матеріалу про змінні та константи. При цьому учні мають навчитися використовувати операцію взяття адреси для змінних та розіменування для вказівників. Одне із застосувань вказівників – це отримання динамічної пам'яті, тому можна запропонувати учням ознайомитися із можливостями виділення та звільнення такого виду пам'яті. Для вивчення матеріалу, пов'язаного із використанням конструктора копіювання та перевантаженням операцій, потрібно оперувати таким поняттям, як "посилання", що було введено у мові С++, як альтернатива вказівникам. Тому після ознайомлення із вказівниками, можна пояснити учням, що таке посилання (див. [2, 54], [3, 23]).

Після цього можна переходити до ознайомлення із засобами високорівневого введення-виведення даних. Тут варто пояснити суть роботи довільної програми чи підпрограми: отримання даних (введення), опрацювання даних (реалізація алгоритму задачі), отримання результату (виведення).

Потрібно ознайомити учнів з певною бібліотекою, яка використовуватиметься у подальшому для реалізації команд введення-виведення. Якщо вчитель обирає для вивчення бібліотеку *stdio* ([3, 121], [1, 176]), то необхідно пояснити загальний синтаксис функцій для виконання вказівок введення-виведення з цієї бібліотеки, навести відповідні приклади.

На наступному етапі варто узагальнити з учнями попередній матеріал, з'ясувати, що називають оператором та лінійним оператором. Потрібно пояснити учням особливості застосування арифметичних операцій, познайомити із використанням основних математичних функцій бібліотеки *math* [1, 185].

Наступним кроком у навчанні має бути більш детальне ознайомлення учнів із поняттям "підпрограма" та застосуванням підпрограм (функцій) у мові Сі. Учні повинні розуміти доцільність використання функцій, вміти створювати власні функції із різними способами передавання параметрів.

Після ознайомлення із можливостями створення та застосування функцій користувача, можна переходити до вивчення питання про структуровані типи даних мови Сі++. Першим таким типом є структура. Учні мають засвоїти основні прийоми оголошення структур та опису їх екземплярів у головній програмі, виробити вміння здійснювати записування/читання даних із полів структур. При доборі вправ та задач стосовно даної теми варто враховувати попередньо вивчений матеріал, зокрема уміння учнів створювати функції користувача. Таким чином вивчення тем "Створення та застосування функцій користувача" та "Тип даних структура" є пропедевтичним етапом вивчення теми "Поняття класу та об'єкта".

Наступним кроком навчання може бути ознайомлення учнів з поняттями "клас" та "об'єкт". Вчитель має показати учням доцільність поєднання у одному типі даних як власне самих даних, так і функцій їх опрацювання; пояснити, з використанням відповідних аналогій, основні концепції об'єктно-орієнтованого підходу у програмуванні.

При доборі задач корисно здійснити міжпредметні зв'язки з іншими дисциплінами. Так, зокрема, можна запропонувати учням задачі на створення класів деяких математичних та фізичних об'єктів.

1. Розробити клас, у якому оголошуються дані про швидкість руху деякої матеріальної точки, що рухається вздовж кола з радіусом *R*. У класі описати наступні функції:
   а) введення даних про матеріальну точку з клавіатури;
   б) виведення даних про матеріальну точку на екран;
   в) обчислення центробіжного прискорення матеріальної точки.

   У головній програмі створити два об'єкти матеріальних точок *pset1* та *pset2*, заповнити їх даними. Виконати тестування функцій класу для обох об'єктів.
2. Створити клас, у якому оголошуються відомості про довжину та ширину деякого прямокутника (дійсні числа). Крім цього описати для класу такі функції:
   а) введення даних про прямокутник з клавіатури;
   б) виведення даних про прямокутник на екран;
   в) обчислення площі прямокутника;
   г) обчислення довжини діагоналі прямокутника;
   д) обчислення периметра прямокутника.

   У головній програмі створити два об'єкти прямокутників *pr1* та *pr2*, заповнити їх даними. Обчислити кут нахилу діагоналей прямокутників до їх сторін. Обчислити периметри та площі прямокутників, вивести на екран значення їх полів.

Після вивчення учнями основ використання простих та складених (структурованих) типів даних у процесі створення програми та застосування лінійного порядку виконання команд у ній, потрібно розпочати ознайомлювати їх із вивченням та реалізацією мовою Сі вказівок розгалуження та повторення. Існують різні методичні підходи щодо питання з якої теми розпочинати вивчення – з розгалуження чи повторення. Класичним є підхід, при якому спочатку вивчають алгоритмічну структуру розгалуження, та лише після цього повторення. Але у працях Морзе Н.В. досить аргументовано обґрунтовується думка про те, що розпочинати вивчення потрібно з вказівок повторення. Кожна точка зору має право на існування, але у реальному процесі навчання все залежить від вчителя, від того, які аналогії він застосує при поясненні матеріалу, яку систему вправ пропонує для вироблення та закріплення умінь і навичок створення програм нелінійної структури. Поряд з цим, як для вказівок повторення, так і для вказівок розгалуження, є одне спільне поняття – "умова" або "логічний вираз". Тому можна припустити доцільність розпочати вивчення розгалужень та повторень саме з пояснення учням суті поняття умови. На цьому етапі учні мають усвідомити, що під логічним виразом розуміють такий вираз, який може набувати лише двох значень: істинно або хибно (один або нуль, так або ні). Вчитель має дібрати достатню кількість прикладів із повсякденного життя та інших навчальних предметів (зокрема з математики), які допоможуть учням засвоїти це поняття. Далі потрібно ознайомити учнів з поняттям простої умови, навести відповідні приклади. Після цього пояснити необхідність використання складених умов, з'ясувати правила їх створення та записування мовою Сі. Пояснити можливість використання законів Де Моргана для перетворення логічних виразів, що містять операцію заперечення.

Для вироблення навичок записування умов різних типів можна запропонувати учням такі вправи.

1. Нижче наведені умови записати з використанням синтаксису мов Сі/Сі++.
   а) значення змінної *a* не перевищує числа десять;
   б) значення змінної *x* відмінне від нуля;
   в) значення змінної *a* не більше від числа п'ять та одночасно з цим значення змінної с менше від числа вісім;
   г) значення кожної із змінних *a, x, y* не перевищує суми значень двох інших змінних;
   д) вік людини перевищує 18 років та не більший від 65 років;
   е) неправильно, що заробітна плата людини менша від 100 грн. та більша від 5000 грн.;
   є) область визначення функції належить відрізкам від -5 до -1 та від 1 до 5;
   ж) область визначення функції – уся числова пряма, крім точок відрізка від -1 до 1;

з) швидкість тіла не може бути від'ємною та перевищувати 500км/год або прискорення тіла знаходиться у межах від 0.5м/с^2 до 2.7м/с^2;

и) деяке число ділиться на 3 або на 5, але не ділиться на 7 або на 4.

2. Відомо, що *a*=3, *x*=7 та *y*=-2. З'ясувати, які з нижче наведених умов будуть істинними, а які хибними.

а) (a>x);
б) (a+x<=2*y);
в) (a*x+y!=0);
г) (x*x+y*y!=a);
д) (a>1 && x<100);
е) (x+y>0 || a*x<100);
є) (x>100 || y<0);
ж) !(x+y==5);
з) (2*x>3 && 3*y>0 || a!=0);
и) (2*x>3 && (3*y>0 || a!=0));

3. Користуючись законами Де Моргана, позбутися операції заперечення перед дужками нижче вказаних умов.

а) !(x>=0);
б) !(a<10);
в) !(a>=0 && a<=10);
г) !(x>y && y!=0 && z==5);
д) !(x<=10 || x>100);
е) !(a!=b || b==3 || c<=10)
є) !(x>5 && (y<=8 || y>=10));
ж) !((a<5 || a>8) && b!=0);
з) !((a<3 || a>12) && (b<=1 || b>5));
и) !(((a<=3 || a>10) && b!=0) || c>=12);

На наступному етапі навчання, можна обрати для вивчення тему "Оператори розгалуження та вибору". :На питання "Чому вивчення розпочинається не з вказівок повторення?" відповідь така. При аналізуванні задач на використання вказівок повторення, були обрані задачі, у яких передбачається використання вказівок розгалуження. Тому, щоб не відбулося порушення принципу послідовності у навчанні, обрано саме такий порядок вивчення цих тем.

Оскільки у результаті вивчення попередньої теми в учнів має бути сформоване уявлення про поняття логічного виразу, то можна очікувати, що вивчення вказівок розгалуження не викликатиме у них особливих труднощів. Завдання вчителя полягає у доборі відповідних вправ та задач, наведенні аналогій, що сприятимуть засвоєнню матеріалу. Перші вправи можуть стосуватися ситуацій повсякденного життя. Мета таких вправ – шляхом побудови блок-схем алгоритмів їх розв'язування пояснити учням суть вказівки розгалуження. Наступним кроком має бути розв'язування задач з використанням простих умов (логічних виразів). Класичними тут є задачі на визначення більшого і меншого з двох, трьох чисел з та без урахування їх рівності. На прикладі деяких з цих задач можна пояснити учням поняття повного, неповного та вкладеного розгалужень.

При розв'язуванні задач на створення вказівок розгалуження з використанням складених умов потрібно навчити учнів спочатку формулювати розв'язок задачі природною мовою, а потім формально записувати його з використанням конструкцій мови програмування.

Задачі, що стосуються даної теми, можна умовно поділити на дві категорії – ті, що розв'язуються з використанням лише структурного підходу, та задачі, які потрібно розв'язувати із застосуванням об'єктно-орієнтованого підходу. Мета задач першої категорії – виробити у учнів навички записування алгоритмів розв'язування типових задач на використання вказівки розгалуження мовою програмування. Задачі другої категорії призначені для навчання учнів застосовуванню об'єктного підходу при аналізуванні її умови, створенню відповідних класів та їх об'єктів.

До задач першої категорії можна віднести наступні задачі.

1. Користувач вказує власній вік. Якщо він менший 18-ти років, то вивести повідомлення "Ви неповнолітній", інакше повідомлення "Ви повнолітній".
2. Користувач вказує власний вік. Якщо він молодший 18-ти років або старший 27 років, то вивести повідомлення "Ви не військовозобов'язаний", інакше вивести повідомлення "Ви військовозобов'язаний".
3. Створити програму, застосовуючи, яку користувач вказує номер свого зодіакального сузір'я (0 – овен, 1 – тілець, …, 11 – риба) і отримує повідомлення на екран про стихію, до якої відноситься це сузір'я (вогонь, вода земля, повітря).
4. Користувач здійснює депозитний внесок деякої суми у банк. Якщо сума внеску не перевищує 500 грн., то процентна ставка за рік – 14%. Якщо сума внеску більше 500, але менше 2000 грн., то – 18%. Якщо ж сума внеску неменша 2000 грн., то – 20%. Підрахувати прибуток від внеску за один рік.
5. Користувач вказує ціле трицифрове число. З'ясувати, яких цифр у ньому більше – парних чи непарних.
6. Для трьох різних значень аргументів, вказаних користувачем, обчислити значення функції $y = \sqrt{\frac{1}{(x-2)(x+3)}}$
7. Користувач вказує значення довжин трьох відрізків. Якщо з них можна утворити трикутник, то знайти його площу і периметр, інакше вивести на екран значення довжини найбільшого відрізка.

Задачі другої категорії можуть бути такі.

1. Створити клас, що моделює трикутник на площині, заданий довжинами своїх сторін. Передбачити у класі функцію перевірки коректності значень сторін трикутника. Використати її при описуванні функцій введення даних про сторони трикутника, обчислення його площі та периметра. Крім цього у класі мають бути описані наступні функції:
   1) виведення даних про сторони трикутника;
   2) обчислення довжин висот, медіан та бісектрис.

   У головній програмі створити три об'єкти трикутників та вивести дані про:
   1) трикутник із найбільшою площею
   2) трикутник із найбільшим периметром

3) для трикутника із найменшим периметром обчислити значення довжин висоти, медіани, бісектриси.

2. Створити клас з даними про постачання товару на склад із наступним властивостями: унікальний номер товару; рік, місяць, день постачання, кількість одиниць товару, ціну за одиницю. У класі мають бути описані наступні функції:
   1) введення даних про постачання (реалізувати перевірку коректності);
   2) виведення даних про постачання;
   3) обчислення загальної вартості постачання.

   У головній програмі створити три об'єкти даних про постачання товарів та вивести відомості про об'єкт з максимальною загальною вартістю.

При вивченні теми "Оператори повторення" варто почати з мотивації необхідності їх застосування у процесі написання програм. Після цього доцільно розглянути достатню кількість вправ, використовуючи які, учні мають ознайомитися із побудовою блок-схем алгоритмів, що містять вказівку повторення. У процесі ознайомлення із синтаксичними правилами записування вказівок повторення мовою програмування корисно розглянути основні задачі, що розв'язуються з їх використанням. До них, зокрема, можна віднести задачі на обчислення суми чи добутку членів деякого набору чисел, вказаних користувачем. Необхідно передбачити можливі помилки учнів при записуванні вказівок повторення та вказати шляхи їх подолання. Учні мають навчитися обирати тип команди повторення при розв'язуванні відповідних задач (повторення з параметром, пердумовою чи післяумовою).

Для вироблення умінь та навичок практичного застосування вказівок повторення, як і при вивченні вказівок розгалуження, можна пропонувати задачі двох категорій.

Метою розв'язування задач першої категорії, як і при навчанні структурного програмування, є вироблення умінь та навичок складання циклічних алгоритмів з урахуванням специфіки їх реалізації мовою програмування Сі/Сі++.

До задач цієї категорії можна віднести такі.

1. Користувач вказує шестицифрове число. З'ясувати, чи буде це число "щасливим" (сума перших трьох цифр дорівнює сумі останніх трьох цифр).
2. Підрахувати суму усіх додатних чисел із діапазону від *m* до *n*, *m*<*n* (вказує користувач), які діляться або на три, або на п'ять.
3. Підрахувати суму перших *n* членів числової послідовності $\frac{i}{1+i^2}, i = 0, 1, 2, ...$
4. Створити програму для виведення на екран значень функції $y = x^2 + 2x - 7$ на відрізку від *a* до *b*, *a*<*b* з кроком *h*, *0*<*h*<*1*, параметри *a*, *b*, *h* вказує користувач.
5. Для вказаного користувачем цілого числа із діапазону від 10 до 1000000, з'ясувати, яких цифр у числі більше – парних чи непарних.
6. Для вказаного користувачем цілого числа із діапазону від 10 до 1000000, з'ясувати, сума яких цифр числа більша – парних чи непарних.
7. Створити програму для обчислення найбільшого спільного дільника двох цілих чисел.
8. Написати програму для знаходження найменшого спільного кратного двох цілих чисел.
9. Створити програму для перевірки – чи є вказане користувачем число із діапазону від 1 до 1000000 простим.
10. Написати програму для виведення на екран усіх простих чисел із вказаного користувачем діапазону.
11. Для усіх попередніх задач передбачити можливість повторного введення даних та отримання результату без перезапуску програми (використати цикл з післяумовою).

Основна мета задач другої категорії – формування навичок застосування вказівок повторення при створенні класів, що моделюють певні об'єкти матеріального та нематеріального походження (математичні, фізичні моделі, моделі баз даних тощо).

Як задачі другої категорії можна запропонувати наступні завдання.

1. Створити клас цілих чисел з єдиною властивістю для збереження значення цього числа та наступними функціями:
   а) введення числа з клавіатури;
   б) виведення числа на екран монітора;
   в) перевірка, чи є число простим;
   г) знаходження найбільшого спільного дільника двох чисел;
   д) обчислення найменшого спільного кратного двох чисел.

   У головній програмі створити два об'єкти цілих чисел та здійснити тестування функцій класу цих чисел.
2. Розробити клас квадратичної функції виду $y = ax^2 + bx + c$. У класі мають зберігатися відомості про коефіцієнти *a*, *b*, *c*; кінці відрізка x_1, x_2 на якому задана функціональна залежність; крок *h*, з яким здійснюється перехід від одного значення аргументу функції до іншого ($h<|x_2-x_1|/10$). У класі мають бути оголошені та описані наступні функції:
   а) введення/виведення даних про вище перераховані властивості;
   б) обчислення значення функції для довільного аргументу;
   в) виведення на екран монітора значень функції для діапазону аргументів від x_1 до x_2 з кроком *h*;
   г) пошук найбільшого та найменшого значення функції на відрізку x_1, x_2 з кроком *h*.

   У головній програмі створити два об'єкти функцій, заповнити їх даними. Знайти найбільше значення кожної функції на вказаному відрізку для одного і того ж значення кроку *h*. Вивести на екран відомості про ту функцію, найбільше значення якої більше.

Наступною темою навчання програмуванню є тема "Тип даних масив", при вивченні якої інтенсивно використовуються знання з попередніх тем "Оператори розгалуження та вибору" і "Оператори повторення".

Здійснити мотивацію вивчення даної теми, можна шляхом пояснення учням складності опрацювання великої кількості однотипних змінних. Наприклад, пригадати, що визначення більшого з трьох цілих чисел потребує використання двох вказівок розгалуження з простою умовою. Якщо ж потрібно знайти більше з чотирьох, то можна використати три вказівки розгалуження з простою умовою. Тобто кількість таких вказівок на одиницю менша від кількості чисел серед яких потрібно визначити найбільше. Зокрема з цієї причини у мовах програмування передбачено використання спеціального типу даних, застосування якого надає можливість зберігати у одній змінній потрібну кількість однотипних значень.

Далі вчитель має повідомити учням про види масивів (лінійні та багатовимірні, зокрема двовимірні), можливості їх опису, заповнення даними та алгоритмами опрацювання цих даних з використанням вказівок повторення та розгалуження. До основних алгоритмів варто віднести наступні: знаходження суми та добутку елементів масиву, лінійний пошук елемента, що відповідає певному критерію (зокрема найбільший чи найменший), упорядкування елементів, бінарний пошук в упорядкованому масиві. Після цього варто познайомити учнів із можливостями опису та використання масивів, елементами яких є структури чи об'єкти класів. Доцільно також навчити їх створювати динамічні лінійні та двомірні масиви.

Після розв'язування разом з учнями типових задач на використання масивів для закріплення та систематизації знань, умінь та навичок можна запропонувати їм такі практичні завдання.

1. Розробити клас лінійного масиву з N цілих чисел ($N>0$ – константа). У класі має бути описана єдина властивість – масив з N елементів. Також у класі повинні бути описані наступні функції.
   1) введення/ виведення елементів масиву;
   2) обчислення суми елементів масиву;
   3) пошук першого максимального елемента у масиві та його індексу;
   4) пошук першого мінімального елемента масиву та його індексу;
   5) сортування елементів масиву;
   6) бінарний пошук елемента масиву;
   7) підрахунок кількості максимальних елементів масиву;
   8) підрахунок кількості різних елементів масиву

   У головній програмі створити об'єкт класу лінійного масиву та виконати тестування його функцій.
2. Розробити клас двовимірного масиву цілих чисел з M рядків та N стовпців ($M>0$, $N>0$, M, N – константи). У класі має бути описана єдина властивість – масив з M рядків та N стовпців. Крім цього у класі потрібно описати наступні функції.
   1) введення/ виведення елементів масиву;
   2) пошук у масиві першого максимального елемента та його індексів;
   3) пошук першого мінімального елемента та його індексів;
   4) заміна місцями рядків та стовпців квадратного масиву (двовимірний масив називають квадратним, якщо у ньому однакова кількість рядків та стовпців);
   5) обчислення суми діагональних елементів квадратного масиву;
   6) сортування за спаданням кожного рядка масиву;
   7) обчислення суми елементів квадратного масиву, що знаходяться нижче головної діагоналі;
   8) перевірка квадратного масиву на "магічність" (суми елементів вздовж стовпців, рядків та обох діагоналей однакові).

   У головній програмі створити об'єкт класу квадратного масиву та виконати тестування усіх його функцій.
3. Створити клас з даними про людину. У класі мають відображатися такі відомості: ідентифікаційний код, рік народження та заробітна платня, а також функції введення/виведення цих відомостей. Після цього створити клас лінійного масиву із N елементів з даними про N людей ($N>0$ – константа). У класі має бути описана єдина властивість – масив з відомостями про людей. Крім цього мають бути описані наступні функції.
   1) введення/ виведення даних про людей;
   2) пошук та виведення на екран монітора відомостей про людей у масиві, заробітна платня яких не перевищує вказаного значення;
   3) пошук та виведення на екран монітора відомостей про людей у масиві, які народилися не раніше вказаного року;

   У головній програмі створити об'єкт масиву з даними про людей та налаштувати роботу програми таким чином, щоб користувач мав змогу обрати код операції над масивом та виконати цю операцію (введення/ виведення, пошук). Після цього надати можливість користувачеві або продовжити роботу з програмою (знову обрати код операції) або завершити виконання програми.

Додаткові завдання підвищеної складності

1. У клас лінійного масиву цілих чисел додати наступні функції.
   1) Додавання нового елемента за вказаним індексом, при цьому виконати перевірку коректності індексу. Якщо значення індексу від'ємне, то додавати елемент у комірку під номером нуль. У випадку, коли значення індексу більше або дорівнює кількості елементів масиву, додавати новий елемент у кінець масиву.
   2) Вилучення елемента за вказаним індексом. Виконати перевірку коректності індексу.
2. У клас двомірного масиву цілих чисел додати такі функції.
   1) Додавання нового рядка за вказаним індексом. Виконати перевірку коректності значення індексу.
   2) Додавання нового стовпця за вказаним індексом. Здійснити перевірку коректності індексу.
   3) Вилучення рядка за вказаним індексом.
   4) Вилучення стовпця за вказаним індексом.
   5) Заміна місцями рядків та стовпців довільного двовимірного масиву (не лише квадратного).

3. У клас лінійного масиву з відомостями про людей додати наступні функції.
   1) Додавання у кінець масиву нового елементу даних з відомостями про людину.
   2) Вилучення елементу даних з відомостями про людину за вказаним індексом. Здійснити перевірку коректності значення індексу. Якщо індекс вказано некоректно, то вилучення не здійснювати.

Варто зауважити, що при розробці класів динамічних структур даних, зокрема масивів, значну роль відіграє створення функцій конструктора та деструктора, що спрощує створення та використання об'єктів класу. Але на даному етапі учні не володіють відповідним матеріалом, тому у текстах задач вказані зауваження щодо коректного застосування об'єктів динамічних масивів. Задача вчителя полягає у тому, щоб пояснити учням правила використання динамічної пам'яті і можливі наслідки її некоректного виділення-звільнення. Поряд з цим можна повідомити учням, що після ознайомлення з темою "Поняття конструктора та деструктора класу" та створення відповідних функцій використання об'єктів динамічних масивів значно спроститься. Таким чином вчитель виконує мотивацію вивчення матеріалу, з яким учні будуть ознайомлені на інших уроках.

Логічним продовженням теми "Тип даних масив" є ознайомлення учнів із можливостями опрацювання рядкових даних. Як відомо, у мові Сі/Сі++ не існує такого типу даних, як рядок. Під рядком тут розуміють масив символів, у якому останній символ (необов'язково останній елемент масиву) це символ ознаки кінця рядка (символ з кодом "нуль" у таблиці ASSCI-кодів). Існують певні труднощі із використанням рядків у мові Сі/Сі++, які пов'язані з тим, що учень повинен чітко розуміти, що таке вказівник і що таке масив. Тому завдання вчителя – з використанням достатньої кількості прикладів пояснити учням можливості опрацювання рядкових даних. У якості таких прикладів можна запропонувати наступні задачі: визначення довжини рядка, копіювання частини рядка, пошук текстового фрагменту у рядку, вилучення текстового фрагменту з рядка, склеювання рядків, реверсування рядка, переведення дійсного числа у рядок і навпаки. Після розгляду цих прикладів вчитель може ознайомити учнів із деякими функціями бібліотеки *string* для роботи із масивами символів.

Для систематизації знань учнів можна запропонувати їм наступні завдання.

1. Створити клас рядка, у якому має бути описана єдина властивість, віднесена до закритої секції – масив, що складається із 255 символів. Також у класі повинні бути описані наступні функції.
   1) введення символів рядка з клавіатури;
   2) виведення рядка на екран;
   3) – 10) функції визначення довжини рядка, копіювання, вилучення, склеювання, реверсування, переведення рядка у дійсне число і навпаки;
   11) перевірка двох рядків на співпадання (результатом роботи функції має бути 1 у випадку, коли значення індексу символу ознаки кінця рядка для обох рядків співпадає та відповідні символи обох рядків також співпадають, у інших випадках результат виклику функції – 0).
   12) підрахунок кількості входження вказаного слова у рядок

   У головній програмі створити об'єкт класу рядка та виконати тестування його функцій.
2. Користувач вказує з клавіатури певну кількість дійсних чисел, розділених символом пробілу. Кількість символів у введеному тексті не повинна перевищувати 255. Знайти суму вказаних чисел.
3. Із використанням класу, створеного у завданні 1, розв'язати задачу на пошук найдовшого слова у вказаному користувачем реченні. Вважати, що слова у реченні розділяються одним символом пробілу, знаки пунктуації не застосовуються.

Наступним кроком навчання може бути ознайомлення учнів з можливостями опрацювання даних із файлів. Для мотивації вивчення цієї теми вчителю доцільно нагадати учням, що рідко у якій програмі не використовується збереження та читання даних із файлу і навести відповідні приклади. Тому при створенні власних проектів виникає необхідність реалізації у програмі підтримки роботи з файлами. Важливо пригадати, що таке файл, та пояснити суть цього поняття з точки зору програміста, а не лише користувача. У процесі пояснення основних понять та виконання операцій над файлами можна використати певні аналогії роботи із книгою чи зошитом (файловий вказівник, зміщення файлового вказівника, відкривання, зчитування, запис, додавання даних у кінець файлу). Після цього можна ознайомити учнів з однією із бібліотек, що містить функції роботи з файлами.

Для вироблення базових практичних навичок опрацювання даних із файлу можна запропонувати такі задачі.

1. Створити текстовий файл, що складається з одного рядка та містить слова, розділені символом пробілу, знаки пунктуації не застосовувати. Написати програму для підрахунку кількості слів у такому текстовому файлі.
2. Створити програму для записування у текстовий файл додатних цілих чисел, що вводяться користувачем з клавіатури. Критерієм припинення введення має бути введення недодатного числа. Відкрити створений файл, прочитати з нього числа та знайти їх суму. Розв'язати задачу з використанням текстового та двійкового файлів.
3. Написати функцію для генерування випадковим чином із діапазону від 1 до 100 цілочисельної послідовності з вказаною користувачем кількістю членів (менше від 50) та записування її у двійковий файл під вказаним користувачем іменем. Створити функцію для читання з двійкового файлу з вказаною назвою цілих чисел та підрахунку їх суми. Виконати тестування створених функцій у головній програмі.
4. Створити функцію, до якої як параметри передаються імена двох текстових файлів. У тілі функції мають бути описані команди для читання кожного символу з одного файлу та записування його у кінець іншого. Використати цю функцію у головній програмі.
5. Кожен рядок текстового файлу складається з однакової кількості символів. У записі рядка використовуються лише два символи "*" та "+". Написати програму для виведення на екран номера рядка та номера стовпця кожного символу "+". Вважати, що нумерація рядків та стовпців починається з нуля.

Після розв'язування цих задача можна запропонувати учнями додати у класи, розроблені на попередніх уроках, функції читання/записування даних з файлу та здійснити їх тестування

Темою, що стосується роботи з файлами, фактично можна завершити ознайомлення учнів з основами структурного програмування. Матеріал, що залишається для вивчення, стосується об'єктно-орієнтованого підходу у програмуванні.

На даному етапі доцільно познайомити учнів з поняттям конструктора та деструктора, пояснити причини та, у окремих випадках, необхідність їх застосування. Необхідно повідомити, що таке конструктор за замовчуванням, порожній конструктор, конструктори перетворення та копіювання, розглянути відповідні приклади.

Для практичного застосування отриманих знань корисно запропонувати учням додати конструктори та, у разі потреби, деструктори до раніше створених класів. Зокрема, можна дати їм такі задачі.

1. Для класу рядка створити такі конструктори:
   а) порожній конструктор (у тілі конструктора нульовому елементу рядка має бути присвоєно символ ознаки кінця рядка);
   б) конструктор перетворення, параметром якого є вказівник на символьний тип (у тілі конструктора символи, на які показує вказівник, мають бути записані як символи рядка)
   в) конструктор копіювання (у тілі конструктора за посиланням на об'єкт іншого рядка його символи мають бути записані, як символи рядка, для якого викликається конструктор).

   Виконати тестування конструкторів у головній програмі.
2. Створити клас ламаної лінії на площині, що містить дві властивості – кількість точок ламаної та вказівник на масив з даними про координати цих точок. У якості класу точок застосувати клас CPset2D, описаний вище. Також у класі мають бути описані наступні функції:
   1) виділення/ звільнення пам'яті під точки ламаної;
   2) порожній конструктор (надання вказівникові на масив точок значення NULL, надання кількості точок ламаної значення нуль);
   3) конструктор перетворення, формальний параметр якого – кількість створюваних точок ламаної (у тілі конструктора має бути реалізованим процес виділення пам'яті за вказівником на масив точок);
   4) конструктор копіювання;
   5) деструктор;
   6) введення/ виведення даних про точки ламаної;
   7) обчислення довжини ламаної;
   8) записування даних про ламану у двійковий файл;
   9) зчитування даних про ламану з двійкового файлу;

   У головній програмі створити об'єкт ламаної з вказаною користувачем кількістю точок (використати конструктор перетворення) та заповнити його даними. Інший об'єкт ламаної створити, як копію попереднього. Обчислити довжину ламаної. Виконати тестування функцій записування/ читання даних об'єкта ламаної.

Наступною темою вивчення є тема "Успадкування". Тут потрібно пояснити учням причини застосування успадкування та різні моделі його реалізації (одиничне, множинне, кільцеве). Але у шкільному курсі достатньо ознайомити учнів із синтаксисом та використанням одиничного успадкування. Доцільно також навести приклад, у якому показується можливість використання захищеної (*protected*) секції та пояснити різницю між захищеною і закритою (*private*) секціями.

Для вироблення практичних навичок застосування можливостей успадкування, доцільно запропонувати учням такі завдання.

1. Створити клас, що моделює земну фауну, та успадкувати від нього класи птахів і риб, спосіб успадкування – *public*.

   У класі фауни описати властивості, що задають назву істоти (не більше 30-ти символів), її середню довжину та середню масу. Конструктору цього класу повинні передаватися значення вказаних властивостей. Значення за замовчуванням дібрати самостійно.

   Клас птахів додатково має містити відомості про розмах крил птаха та максимальну висоту польоту. Конструктор цього класу має приймати значення усіх властивостей, у тому числі властивостей базового класу. Значення за замовчуванням дібрати самостійно.

   У класі риб додатково має бути описана властивість, що характеризує максимальну глибину занурення. Конструктор класу риби теж має приймати значення усіх властивостей, для яких значення за замовчуванням обрати самостійно.

   У головній програмі створити об'єкти риби та птаха. Значення їх властивостей повинен вказати користувач. Введені дані вивести на екран та зберегти у текстовому файлі.
2. Створити клас довільного трикутника, заданого координатами своїх вершин (об'єкти класу точки) на площині та успадкувати від нього класи прямокутного і рівностороннього трикутників, спосіб успадкування – *protected*.

   Клас довільного трикутника має містити порожній конструктор, за яким створюватиметься трикутник з вершинами (-1, 0), (0, 3) та (1, 0), а також функцію обчислення площі.

   Клас прямокутного трикутника додатково містить дві властивості, що задають довжини його катетів. До конструктора класу прямокутного трикутника передаються вершина прямого кута і довжини катетів. Передбачити для цих параметрів значення за замовчуванням: вершина – точка (0, 0), довжини катетів – 1. У конструкторі потрібно розрахувати координати двох інших вершин, якщо відомо, що катети паралельні координатним осям. Також даний клас повинен містити функції для зміни довжин катетів та координат вершини прямого кута (три функції). У тілі цих функцій має здійснюватися переобрахунок координат точок трикутника.

   Клас рівностороннього трикутника додатково містить властивість, що визначає довжину його сторони та конструктор. У якості параметрів до конструктора мають передаватися координата однієї

з вершин та значення довжини сторони трикутника. Передбачити для цих параметрів значення за замовчуванням: вершина – точка (0, 0), довжина сторони – 1. Як і для класу прямокутного трикутника, створити функції зміни координат вершини та довжини сторони трикутника.

У головній програмі створити об'єкти кожного з трикутників описаних класів. Для прямокутного трикутника збільшити кожен з катетів на вказане користувачем значення. Для рівностороннього трикутника збільшити довжину сторони на вказане користувачем значення. Обчислити площу утворених трикутників. Результати вивести на екран та зберегти у текстовому файлі.

Логічним продовженням теми "Успадкування" є вивчення теми "Перевизначені функції класу. Віртуальні функції". Ознайомлення з новим матеріалом можна розпочати з мотивації доцільності використання для класів деякого ієрархічного ланцюжка однакових імен функцій, у яких здійснюється однотипне опрацювання даних. Далі, використовуючи приклади, потрібно показати учням зручність застосування віртуальних функцій, принципи раннього та пізнього зв'язування. Також потрібно пояснити, з відповідною аргументацією, використання понять "абстрактна функція" та "абстрактний клас".

Щоб надати можливість учням на практиці застосувати теоретичні знання з даної теми, можна запропонувати їм такі завдання.

1. Оголосити клас з даними про людину. До властивостей цього класу віднести ім'я (не більше 30-ти символів) та рік народження. У класі мають бути оголошені такі віртуальні функції:
   1) дві функції записування/читання даних про людину до/з текстового файлу (відповідним функціям, як параметр, має передаватися файловий вказівник);
   2) деструктор.

   Від класу з даними про людину успадкувати класи робітника та інженера. У цих класах повинні бути оголошені та описані перевизначені функції базового класу.

   Клас робітника додатково має містити властивості для збереження даних про номер його розряду та кількість опанованих професій.

   У класі інженера повинні зберігатися дані про назву ВУЗу, який він закінчив, та назву спеціальності (не більше 250 символів).

   Створити текстовий файл, перший рядок якого містить кількість блоків даних про людей, а інші рядки містять дані про цих людей – у кожному рядку відомості про одну людину.

   Формат рядка з даними про робітника такий:
   1) ціле число, що вказує на код професії (1 – робітник, 2 – інженер);
   2) ім'я робітника та його рік народження;
   3) номер розряду робітника;
   4) кількість опанованих професій.

   Формат рядка з даними про інженера наступний:
   1) ціле число, що вказує на код професії (1 – робітник, 2 – інженер);
   2) ім'я інженера та його рік народження;
   3) назва ВУЗу, у якому навчався інженер;
   4) назва спеціальності інженера.

   Для спрощення читання з файлу текстових даних (ім'я, назва вузу, назва спеціальності) слова у цих даних розділяти символом підкреслення, а окремі дані – символом пропуску. Так, наприклад, дані про інженера у файлі можна зберігати наступним чином:

   2 Іваненко_Петро 1982 КПІ технології_та_матеріалознавство

   У головній програмі відкрити створений файл та створити відповідний динамічний масив з даними про робітників та інженерів. Знайти у такому масиві відомості про наймолодшу людину і вивести їх на екран монітора та зберегти у текстовому файлі. Звільнити пам'ять динамічного масиву об'єктів даних про людей.
2. Створити абстрактний клас плоскої геометричної фігури, що містить єдину властивість – назву фігури (не більше 30-ти символів), а також наступні віртуальні функції:
   1) дві функції введення/виведення даних про фігуру (клавіатура, монітор);
   2) дві функції записування/читання даних фігури до/з текстового файлу (відповідним функціям, як параметр, має передаватися файловий вказівник);
   3) обчислення периметру фігури;
   4) обчислення площі фігури;
   5) деструктор.

   Від класу плоскої геометричної фігури успадкувати класи трикутника, паралелограма та круга. Усі похідні класи мають містити перевизначені функції базового класу.

   Як властивості класу трикутника, використати довжини двох його суміжних сторін та кут між ними у градусах. Цей клас має містити функцію обчислення довжини третьої сторони.

   За властивості класу паралелограма обрати довжини його суміжних сторін та кут між ними у градусах.

   У класі круга, крім успадкованої властивості "назва фігури", додатково створити властивість, що позначає довжину його радіуса.

   У головній програмі запропонувати користувачеві вказати загальну кількість створюваних фігур потім надати можливість вказати тип кожної фігури (трикутник, паралелограм, круг) та її властивості (використати динамічний масив вказівників на об'єкти базового класу). Після створення усіх фігур спочатку вивести відповідні дані на екран та до файлу у порядку зростання периметра, а потім – у порядку зростання площі. Звільнити пам'ять динамічного масиву об'єктів даних про геометричні фігури.

Наступною темою вивчення основ об'єктно-орієнтованого програмування може бути тема "Дружні" функції та класи. Попереднє оголошення заголовків класів". Вивчення такої теми варто розпочати з пояснення доцільності використання "дружніх" функцій і класів для випадку, коли властивості класів віднесені до закритої чи захищеної секції. Потрібно пояснити, що при цьому може

виникнути необхідність у попередньому оголошенні заголовків класів та показати, як здійснюється таке оголошення.

Для вироблення практичних навичок використання "дружніх" функцій та класів, учні мають розв'язати наступні задачі.

1. Створити клас банківського рахунку. Властивості – тип рахунку (звичайний чи валютний, не більше чотирьох символів), кількість грошей на ньому та унікальний код рахунку (шестизначне число) розташувати у закритій секції. Конструктору рахунка мають передаватися значення усіх його властивостей, крім унікального коду.

   Створити клас клієнта, "дружній" класу банківського рахунку. У класі клієнта створити три закриті властивості: ім'я клієнта (не більше 30-ти символів), вказівник на об'єкт його рахунку, код клієнта (шестизначне число). Конструктор цього класу, як параметр, має приймати вказівник на певний рахунок та генерувати унікальний код клієнта, значення якого потрібно надати унікальному коду рахунка цього клієнта. Тобто, код клієнта і код рахунку – це ті властивості, за якими встановлюється належність деякого рахунку певному клієнтові. Крім цього у класі клієнта створити функцію виведення на екран відомостей про клієнта і його рахунок.

   Створити глобальні, "дружні" до вище описаних класів, функції поповнення/зняття грошей на рахунку, яким, як параметри, передаються посилання на об'єкти рахунку та клієнта і сума, на яку потрібно здійснити поповнення/зняття. У тілі цих функцій потрібно проаналізувати, чи володіє вказаний клієнт заданим рахунком, і лише після цього виконувати операції над грошовою сумою.

   В головній програмі створити два об'єкти рахунків та відповідні їм об'єкти клієнтів. У подальшому організувати інтерфейс програми таким чином, щоб користувач (наприклад, операціоніст банку) мав змогу обрати код операції (поповнення, зняття чи вихід з програми), код клієнта і рахунок. У випадку, коли обрані рахунок та код клієнта не відповідають один одному – виводити на екран повідомлення про цей факт. Якщо ж така відповідність існує, то інформувати користувача про здійснення обраної ним операції. Наприкінці виконання програми вивести на екран дані про клієнтів та їхні рахунки.
2. Створити клас космічного зонду і космічного корабля.

   Клас космічного зонду повинен містити наступні закриті поля: назва зонду; поле для збереження даних, зібраних зондом (не більше 255 символів); вказівник на космічний корабель, який обмінюється даними із зондом. Конструктору цього класу, як параметр, передається назва зонду. Використання інших функцій не передбачається.

   У класі космічного корабля мають бути описані такі закриті поля: назва космічного корабля (не більше 30-ти символів); база з даними, зібраними з використанням зондів (двовимірний масив символів не більше, ніж з 10-ти 255-символьних рядків, кожен рядок – це набір даних, прийнятих від зонда); номер рядка бази, у який можна записувати дані; вказівник на зонд, яким керує корабель. Конструктору такого класу, як параметр, передається ім'я космічного корабля. Крім конструктора, у цьому класі описати функцію для виведення на екран вмісту бази даних космічного корабля.

   Описати такі глобальні функції.

   1) Передавання від космічного корабля до зонду команди на збирання даних з певного ресурсу. У цій функції передбачити, що зонд не може прийняти таку команду, якщо у ньому вже є зібрані, але не відіслані дані. Як ресурс даних використовувати довільний текстовий файл.
   2) Передавання від космічного корабля до зонду команди на пересилання даних на корабель. У такій функції передбачити, що із зонду не можуть передаватися порожні дані та дані, що не поміщаються до бази космічного корабля.
   3) Функція виведення на екран назви зонда, який виконує команду запиту вказаного космічного корабля.

   У головній програмі створити три об'єкти космічних кораблів та чотири об'єкти зондів. Налаштувати інтерфейс програми таким чином, щоб у користувача була можливість обрати довільні космічний корабель і зонд, за їх порядковими номерами, та виконати одну із чотирьох команд:

   1) передавання команди від космічного корабля до зонду на збирання даних;
   2) передавання зібраних даних від зонду до космічного корабля;
   3) виведення для обраного корабля назви зонду, з яким він здійснює обмін даними;
   4) виведення на екран бази даних обраного корабля.

   Після виконання однієї з вище перерахованих команд надати користувачеві можливість або продовжити роботу з програмою (знову обрати корабель, зонд та команду), або завершити її виконання.

   Зауваження. Як ресурси даних можна створити кілька текстових файлів з описом планет сонячної системи, комет, астероїдів тощо.

Наступною темою для вивчення є тема "Перевантаження стандартних операцій". При ознайомленні учнів з цією темою варто показати зручність застосування зокрема основних арифметичних операцій не лише для стандартних типів мови Сі, але і для об'єктів класів. На етапі повідомлення нового матеріалу можна побудувати разом з учнями, наприклад, клас вектора на площині, клас комплексного числа і т. д. На цих прикладах зручно пояснити доцільність створення глобальної функції перевантаження операції, що виконується над об'єктами різних типів, зокрема при множенні дійсного числа на вектор.

Для набуття практичних навичок розв'язування задач, де потрібно використовувати перевантаження стандартних операцій для створюваних класів, можна запропонувати учням такі завдання.

1. Створити клас-оболонку над цілочисельним типом (*int*) мови Сі. У класі мають бути дві закриті властивості: поле для збереження цілочисельного значення та повідомлення про результат виконання однієї з арифметичних операцій. Створити наступні функцій класу.
   1) Конструктор за замовчуванням, а також конструктори копіювання та перетворення (конструктори за замовчуванням та перетворення можна описати як один).

2) Функція введення даних про число з клавіатури. Такій функції мають передаватися наступні параметри: мінімальне та максимальне значення діапазону, з якого потрібно ввести число; повідомлення-запрошення для введення; повідомлення про некоректність введення значення. У тілі функції передбачити можливість повторного введення даних у випадку їх некоректного введення. Для усіх параметрів цієї функції передбачити значення за замовчуванням.
3) Функція виведення даних на екран монітора. Для такої функції передбачити єдиний параметр-повідомлення, що виводиться перед виведенням цілочисельного значення.
4) Функції збереження/завантаження даних класу в/із файл/а. Як параметр, таким функціям має передаватися файловий вказівник.
5) Функції перевантаження арифметичних операцій над цілочисельними даними (+, -, /, *, %, унарний мінус). Для певних операцій у тілі відповідних функцій передбачити можливість переповнення. У разі переповнення повинно формуватись відповідне повідомлення, що має бути збереженим у спеціально створеній для цього властивості класу.
6) Функція виведення на екран значення властивості-повідомлення про помилку виконання арифметичної операції.

У головній програмі створити три об'єкти цілих чисел і налаштувати інтерфейс програми таким чином, щоб користувач отримав можливість вказати значення для двох з них (з клавіатури або текстового файлу), обрати потрібну арифметичну операцію, результат її виконання зберегти у третьому об'єкті та вивести на екран монітора. У випадку переповнення виводити на екран відповідне повідомлення. Наприкінці роботи програми зберегти дані усіх об'єктів у текстовому файлі.

2. Розробити клас прямокутника із суміжними сторонами, паралельними координатним осям, що визначається на площині дійсними координатами верхнього лівого та нижнього правого кутів. Для класу створити наступні функції.
   1) Конструктор за замовчуванням. За цим конструктором має створюватися квадрат (0; 1) – (1, 0).
   2) Функція введення з клавіатури даних про прямокутник.
   3) Функція виведення на екран монітора даних про прямокутник.
   4) Функції збереження/завантаження в/із файл/у даних про прямокутник.
   5) Функції обчислення площі та периметра прямокутника.
   6) Операція додавання прямокутників. Сумою двох прямокутників вважається прямокутник, ліва сторона якого проходить вздовж лівої сторони прямокутника-доданка з меншою абсцисою; права сторона проходить вздовж правої сторони прямокутника-доданка з більшою абсцисою; верхня сторона проходить вздовж верхньої сторони прямокутника-доданка з більшою ординатою; нижня сторона проходить вздовж нижньої сторони прямокутника-доданка з меншою ординатою.
   7) Операція множення прямокутників. Результатом множення двох прямокутників називається прямокутник, що утворюється, як перетин двох прямокутників-множників. Врахувати випадок, при якому два прямокутники не мають спільної частини.

   У головній програмі створити три об'єкти прямокутників. Налаштувати інтерфейс програми так, щоб у користувача була можливість вказати дані двох об'єктів прямокутників з клавіатури або файлу, обрати операцію, яку потрібно виконати, результат виконання операції у третьому об'єкті та вивести на екран монітора. Якщо результат виконання операції існує, то для отриманого прямокутника обчислити площу та периметр і вивести їх значення на екран монітора. Наприкінці роботи програми дані усіх об'єктів прямокутників зберегти у файлі.
3. Створити клас, що моделює множину, яка складається із літер латинського алфавіту. Елементи, які зберігатимуться у об'єкті множини, не можуть повторюватися. В такому класі передбачити дві закриті властивості – масив для збереження елементів та їх кількість у множині. Крім цього до класу мають увійти наступні функції.
   1) Конструктор за замовчуванням – створюється об'єкт порожньої множини.
   2) Конструктор перетворення з цілочисельним параметром в межах від 1 до 26. У результаті використання цього конструктора має бути створена множина із вказаною кількістю літер, починаючи від літери "A" і закінчуючи літерою, номер якої в алфавіті відповідає вказаному цілому значенню.
   3) Конструктор копіювання.
   4) Функція введення елементів множини з клавіатури. Врахувати, що у множині не може бути двох однакових елементів.
   5) Функція виведення елементів множини на екран.
   6) Функції збереження/завантаження властивостей класу у/із файл/а. Як параметр, таким функціям має передаватися файловий вказівник.
   7) Перевантажена операція надання значення від одного до іншого об'єкта множини.
   8) Перевантажені операції додавання, віднімання, множення та унарний мінус для об'єктів класу. Правила виконання операцій такі.
      а) Сумою двох множин називають таку множину, що складається з елементів, які належать або першій або другій множині.
      б) Різницею множини *A* та *B* називають таку множину, яка складається з елементів, що належать множині *A* і не належать множині *B*.
      в) Добутком двох множин називають таку множину, що складається з елементів, які належать і першій і другій множині.
      г) Результатом виконання операції унарного мінуса для множини *A* є така множина *B*, що складається лише з тих елементів, що не належать до множини *A*.

У головній програмі запропонувати користувачеві створити три об’єкти множин, для двох з них вказати значення з клавіатури або текстового файлу. Налаштувати інтерфейс програми таким чином, щоб користувач міг обрати потрібну операцію та результат її виконання зберегти у третьому об’єкті і вивести на екран монітора. Наприкінці роботи програми дані усіх об’єктів зберегти у текстовому файлі.

Завершити вивчення основ об’єктно-орієнтованого програмування можна ознайомленням з темою "Створення і використання шаблонів функцій та класів". При вивченні цієї теми потрібно показати учням зручність використання шаблонів. У результаті учні повинні розуміти, що застосування шаблонів спрощує опрацювання різнотипних даних за одними і тими ж алгоритмами, вміти створювати власні шаблони функцій та класів.

Щоб учні на практиці мали можливість навчитися створювати шаблони та використовувати їх у власних програмах, можна запропонувати їм наступні завдання.

1. Створити шаблони функцій для пошуку максимального та мінімального елементів у масиві із *n* елементів (об’єктів) довільного типу (класу). У головній програмі виконати тестування цих функцій для цілочисельного масиву та масиву ламаних ліній (використати клас ламаної лінії, описаний раніше).
2. Створити клас шаблон динамічного масиву елементів довільного типу. У класі передбачити функції додавання/вилучення рядків та стовпців а також функцію пошуку найменшого серед найбільших елементів кожного рядка . У головній програмі виконати тестування функцій розробленого класу, створивши масив елементів дійсного типу та масив трикутників. Для порівняння об’єктів трикутників використати наступне правило: "з двох трикутників більшим є той, площа якого більша".

ЛІТЕРАТУРА

1. С.О. Бочков, Д.М. Субботин Язык программирования Си для персонального компьютера. – М.: Радио и связь, 1990. – 384 с.
2. Майкл Дж. Янг Visual C++ 6. Полное руководство: Пер. с англ. – К.: Издательская группа BHV, 2000. – 1056 с., ил.
3. Мейерс С. Наиболее эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов: Пер. с англ.– М: ДМК Пресс, 2000.– 304 с.: ил. (Серия "Для программистов").
4. Морзе Н.В. Основи інформатики. Екзаменаційні білети: запитання та відповіді .– К.: ДиаСофтЮП, 2000. – 160 с.