Лабораторні роботи

1. Лабораторна робота №1. Використання стандартної бібліотеки шаблонів STL

2. Лабораторна робота №2. Використання динамічно створених компонентів

3. Лабораторна робота №3 (для магістрів). Створення компонентів користувача

4. Лабораторна робота №3 (для спеціалістів). Використання графіки

5. Лабораторна робота №4. Доступ до баз даних

1. Лабораторна робота №1. Використання стандартної бібліотеки шаблонів STL

Побудувати проект SDI з використанням стандартної бібліотеки шаблонів STL.

Для цього

- вхідні дані з елементів керування форми занести у зручні для реалізації задачі контейнери, утворені засобами бібліотеки шаблонів STL (не обов'язково використовувати саме запропоновані у варіантах завдань контейнери);
- при розв'язуванні задачі використати ітератори і алгоритми бібліотеки STL;
- передбачити три командні кнопки, по яких елементи керування форми заповнюються підготовленими тестовими прикладами.

Інтерфейс програми повинен бути зручним і зрозумілим довільному користувачу. У програмі при можливості передбачити контроль вводу користувача. Елементи керування, які тимчасово не використовуються, зробити недоступними або невидимими.

Варіанти завдань

1. Детермінований скінченний автомат вводиться за допомогою текстової таблиці TStringGrid. Зчитати його у пам'ять. Представлення в пам'яті – у вигляді масиву з елементами вигляду

стан, ознака заключності, символ переходу, стан після переходу

Для реалізації використати зручний контейнер, можливо, такий – *vector<pair<pair<char,boolean>, <pair<char,char>* >. За допомогою утвореного представлення визначити, чи належить введений ланцюжок мові автомата. Вивести отриману таблицю переходів у багаторядкове поле редагування TMemo.

2. Недетермінований скінченний автомат вводиться за допомогою TStringGrid. Зчитати його у пам'ять. Представлення у пам'яті – у вигляді масиву з елементами вигляду

стан, символ переходу, рядок станів після переходу

Для реалізації використати зручний контейнер, можливо, такий – *vector<pair<pair<char,char>,string>*. Вивести отриману таблицю переходів у багаторядкове поле редагування TMemo. За допомогою утвореного представлення реалізувати алгоритм вилучення недосяжних станів автомата.

3. У багаторядковому полі редагування TMemo задано праволінійну граматику. Кожному нетерміналу граматики відповідає один рядок TMemo. Правила у рядку розділяються символом "|". Зчитати граматику у пам'ять. Представлення у пам'яті – у вигляді асоціативного масиву з елементами вигляду

нетермінал, правила

Для реалізації використати контейнер вигляду *map<char, string>*. Ключем у асоціативному масиві є нетермінал. З утвореного представлення утворити новий контейнер вигляду *vector<<pair<char, string>* . У рядок входить тільки одне правило (немає символів "|", а нетермінали у масиві можуть повторюватись). Вивести перетворену граматику у список TListBox. Перевірити, чи дійсно граматика праволінійна.

4. У багаторядковому полі редагування TMemo задано ліволінійну граматику. Кожному нетерміналу граматики відповідає один рядок TMemo. Правила у рядку розділяються символом "|". Зчитати граматику у пам'ять. Представлення у пам'яті – у вигляді асоціативного масиву з елементами вигляду

нетермінал, правило1, правило2,…

Для реалізації використати контейнер вигляду *map<char, vector<string> >*. Ключем у асоціативному масиві є нетермінал. Вивести множини нетерміналів і терміналів (контейнер *set<char>*). Згенерувати допустимий ланцюжок граматики, випадковим чином вибираючи правила для нетерміналів. Перевірити, чи дійсно граматика ліволінійна.

5. У багаторядковому полі редагування TMemo задано неорієнтований граф у вигляді інцидентних вершин

вершина 1 список інцидентних вершин

вершина 2 список інцидентних вершин

...

вершина n список інцидентних вершин

Зчитати його у пам'ять. Представлення у пам'яті – у вигляді масиву списків інцидентних вершин (контейнер *vector<list<int> >*). Ввести номери двох вершин графу і вивести шлях, по якому від першої вершини можна попасти у другу.

6. Неорієнтований граф заданий у вигляді списків інцидентних вершин (див. варіант 5). Зчитати його у пам'ять. Представлення в пам'яті – у вигляді масиву черг інцидентних вершин (контейнер *vector<queue<int> >*). Задано номер вершини. Увести додаткову нумерацію вершин: заданій вершині присвоїти номер 0, всім інцидентним з нею вершинам (шар 1) –

номер 1, всім ще не переглянутим вершинам, інцидентним вершинам з номером 1 (шар 2) – номер 2 і т.д. Для цього зберігати і демонструвати на кожному кроці список вершин поточного шару. Вивести вершини по шарах.

7. У багаторядковому полі редагування TMemo записані слова через пробіл. Є деяка множина ключових слів, збережених в полі редагування TEdit. Створити таблицю зустрічальності для неключових слів у вигляді

слово (рядок, номер слова у рядку), ... , (рядок, номер слова у рядку)

і зберегти її у другому TMemo. Для збереження конструкцій виду *(рядок, номер слова у рядку)* використати контейнер *vector<pair<string,int> >*.

8. Розробити структуру даних «словник, поповнюваний словами з файлів». У словнику зберігати слово й кількість його повторень. Словник (контейнер *vector<pair<string,int> >*) повинен уміти себе зберігати у файл, відновлювати себе з файлу у багаторядкове поле редагування TMemo й видавати у другому TMemo або слова, упорядковані за алфавітом, або слова, упорядковані за по зустрічальності. Заголовок мітки, який позначає результат, змінювати в залежності від радіокнопки.

9. У багаторядковому полі редагування TMemo записані слова. Є таблиця синонімів, записана в іншому TMemo (для одного слова може бути кілька синонімів). Розробити структуру даних, що підтримує таблицю синонімів. Замінити кожне слово у першому TMemo на випадковий синонім і результат записати в третє TMemo. Використати зручний для використання контейнер бібліотеки STL, можливо такий – *map<string, vector <string> >*.

10. Є список студентів, що вивчають математику, фізику, біологію й хімію, заданий у файлі у вигляді рядків

прізвище, предмет

Зчитати файл у пам’ять. Представлення в пам'яті – контейнер *vector<pair<string, string> >*. Кожний предмет вивчає деяка кількість студентів. Розробити структури даних, що дозволяють відповідати на запити виду «Видати прізвища всіх студентів, що вивчають математику й фізику, але не вивчають програмування». Питання формуються за допомогою елементів керування типу TCheckBox, а результати виводяться у багаторядковому полі редагування TMemo.

11. Із заданої послідовності натуральних чисел утворити послідовність ключів (натуральних чисел, які не повторюються). Побудувати хеш-таблицю зі списками (використати контейнер *vector<stack<int> >*). Розмірність хеш-таблиці вводиться, а хеш-функція вибирається радіокнопками. У текстову таблицю TStringGrid вивести утворену хеш-таблицю.

12. Ввести координати точок на площині. Впорядкувати точки в залежності від радіокнопок або за відстанню від початку координат, або за зростанням кута між векторами з заданими координатами та віссю абсцис (використати контейнер *vector<pair<double, pair<int,int> >*). При необхідності (використати TCheckBox) знайти відстані і/або кути. Області виводу, які не використовуються, зробити невидимими.

13. Із заданого тексту (контейнер *vector<string>*), який складається з окремих рядків, утворити текст з такими ж рядками, в яких залишити або всі голосні літери, або всі цифри, або всі

символи без знаків пунктуації, які були у рядках (використати радіокнопки). Заголовок мітки, який позначає результат, змінювати в залежності від радіокнопки.

14. Ввести два масиви (контейнер *vector<float>*). Упорядкувати їх за спаданням або зростанням у залежності від ознаки (використати радіокнопки). При необхідності (використати ознаку) – злити в один упорядкований масив і знищити повтори елементів.

15. Дано масив (контейнер *vector<int>*) і номер позиції, з якої його треба перетворити. Впорядкувати масив у залежності від ознаки (використати радіокнопки) або в порядку зростання, або в порядку спадання, або реверсувати масив. Перенести перетворену частину на початок масиву.

16. Ввести натуральне число *n*. Вивести в різні списки (контейнер *list<int>*) взаємно прості з числом *n* числа, які менші за *n*; прості числа, які менші за *n*; додатні дільники числа *n*. Виводити результати, якщо включено відповідну ознаку. При вводі нового числа *n* очистити списки.

17. Перетворити введений масив чисел (контейнер *vector<int>*), зсунувши його циклічно на *k* позицій (*k* вводиться за допомогою форми DELPHI) вліво або вправо (в залежності від радіокнопки). Вивести перетворений масив в список.

18. У заданому тексті підрахувати і при необхідності вивести слова, які починаються і закінчуються однією буквою; містять тільки три букви; є симетричними. Знайдені групи слів зберігати у контейнерах типу *vector<string>.* Упорядковані групи слів виводити в три різні області в залежності від заданої ознаки.

19. Дано натуральне число *n*, дійсні числа $y_1, \ldots, y_n$

(використати контейнер *vector<float>*). Обчислити одну з величин:

- суму всіх елементів, що належать відрізку [1,5];
- кількість від’ємних елементів;
- кількість елементів, що не належать відрізку [3,8].

Потрібний результат отримати при виборі відповідної радіокнопки. Використати одну область для виведення результату. Назву області змінювати.

20. Дано цілі числа $a_1,\ldots,a_n$ (використати контейнер *vector<int>*). Одержати:

- кількість непарних і від'ємних;
- кількість елементів, що задовольняють умові ;
- суму чисел, кратних 5.

Потрібний результат отримати при виборі відповідного елемента типу TCheckBox. Утворити 3 області для виведення результатів. Області результатів, які не використовуються, зробити невидимими.

21. Дано рядок символів. Вивести або всі слова, в яких є однакові букви, або кількість слів, в яких усі букви різні (використати радіокнопки). Для цього застосувати контейнери *vector<string> і set<char>.* Назву вікна результатів міняти в залежності від інформації. Вивести повідомлення, якщо потрібних слів немає (використати форму DELPHI).

22. Дано рядок з слів і окреме слово. Вивести або всі слова, в яких є всі букви цього слова, або кількість слів, в яких немає жодної букви введеного слова (використати радіокнопки). Для цього застосувати контейнери *vector<string> і set<char>.* Назву вікна результатів міняти в залежності від інформації. Вивести повідомлення, якщо потрібних слів немає (використати форму DELPHI).

23. У полях редагування TEdit ввести три множини X1, X2, X3, що містять цілі числа (використати контейнер *set<int>*). Відомо, що потужність кожної із цих множин більша 10. Сформувати нову множину Y=(X1∪X2) (X2\X3), з якої виділити підмножину непарних чисел. Вивести вихідні й отримані множини, а також проміжні результати, якщо включено відповідну ознаку TCheckBox.

24. У полях редагування TEdit ввести три множини X1, X2, X3, що містять цілі числа (використати контейнер *set<int>*). Відомо, що потужність кожної із цих множин більша 10. Сформувати нову множину Y=(X1∪X2)\(X2×X3) і вивести її потужність. Побудувати множину Z з чисел, які в множині Y і діляться на 6 без остачі. Вивести вихідні й отримані множини, а також проміжні результати, якщо включено відповідну ознаку TCheckBox.

25. Задано послідовність одно-, дво- і тризначних натуральних чисел у полі редагування TEdit (використати контейнер *vector<int>*). Надрукувати множину цифр (використати контейнер *set<int>*), яка відповідає на запиту виду «Видати всі цифри, що входять у запис тризначних чисел і не входять у запис двозначних». Питання формуються за допомогою елементів керування типу TCheckBox.

26. Задано послідовність натуральних чисел у полі редагування TEdit (використати контейнер *vector<int>*). Сформувати множини цифр (використати контейнер *set <char>*)

- що входять у запис кожного числа;
- що входять у запис парних чисел і не входять запис непарних;
- що входять у запис хоча б одного числа більше одного разу.

Потрібний результат отримати при виборі відповідного елемента типу TCheckBox.

27. Задано непорожню послідовність слів з малих латинських літер. Між сусідніми словами – пробіли. Сформувати масив слів (контейнер *vector<string>*) і вивести його у список TList. Також вивести в алфавітному порядку множини літер (використати контейнер *set<char>*)

- букви, що входять до найдовших слів;
- голосні букви, що входять не більше, ніж до одного слова;
- букви, що входять до слів, які закінчуються голосною літерою.

Потрібний результат отримати при виборі відповідної радіокнопки. Використати одну область для виведення результату. Назву області змінювати.

28. Задано непорожню послідовність слів з малих латинських літер. Між сусідніми словами – кома. Сформувати масив упорядкованих слів (контейнер*vector*<*string*>) і вивести його у TMemo. Також вивести в алфавітному порядку множини літер (використати контейнер *set* <*char*>)

- дзвінкі приголосні букви, що входять тільки до одного слова;
- приголосні букви, що не входять до жодного слова;
- приголосні букви, що входять до слів, які починаються дзвінкою приголосною.

Потрібний результат отримати при виборі відповідного елемента типу TCheckBox.

29. Задано непорожню послідовність слів з малих латинських літер. Між сусідніми словами – знаки пунктуації. Сформувати масив слів (контейнер*vector*<*string*>) і вивести його у список TList. Також вивести в алфавітному порядку множини слів (використати контейнер *set* <*string*>)

- що мають парну кількість букв;
- які є симетричними;
- що складаються з різних символів.

Потрібний результат отримати при виборі відповідної радіокнопки. Використати одну область для виведення результату. Назву області змінювати.

30. Задано непорожню послідовність слів. Кожне слово – послідовність цифр. Між сусідніми словами – знаки пунктуації. Вивести множини слів (використати контейнер *set* <*string*>)

- що є щасливими (мають однакову суму цифр в кожній з половин слова);
- які складаються з цифр, що є простими числами;
- які складаються з парних цифр.

Сформовані множини слів вивести у списках TList.

Потрібний результат отримати при виборі відповідного елемента типу TCheckBox.

2. Лабораторна робота №2. Використання динамічно створених компонентів

Побудувати Windows-додаток з багатодокументним інтерфейсом (MDI). Для цього

- перетворити звичайну форму (FormStyle = fsNormal) з лабораторної роботи №1 у дочірню форму (FormStyle = fsMDIChild);
- створити батьківську форму (FormStyle = fsMDIForm) з декількома пунктами меню (наприклад, File, Windows, About), в тому числі з пунктом About, за допомогою якого викликається форма з умовою задачі;
- зробити батьківську форму головною формою проекту;
- на дочірній формі використати динамічно створені компоненти і динамічне переключення обробників подій (для цього при необхідності доповнити умову задачі);
- підключити до проекту форму, побудовану в середовищі Delphi, в якій розмістити незначну частину розробки (наприклад – умову задачі);
- у різних екземплярах дочірньої форми, які динамічно створюються при виборі відповідних елементів підменю Windows, підготувати тестові приклади для демонстрації роботи програми.

3. Лабораторна робота №3 (для магістрів). Створення компонентів користувача

Створити компонент користувача із вказаними властивостями методами і подіями (інших подій, крім заданих, компонент не повинен мати). В якості пращура краще брати схожий абстрактний клас. Протестувати створений компонент у власному проекті до інсталяції його на палітру. Інсталювати компонент на сторінку Samples палітри компонентів. Випробувати інстальований компонент.

Варіанти завдань

1. Компонент – заокруглений прямокутник з надписом. Властивості компонента – розміри прямокутника, текст надпису, колір фону, колір тексту. Методи – зміна тексту, обертання тексту. Подія – OnClick.

2. Компонент – кільце. Властивості компонента – зовнішній радіус, ширина кільця, колір фону, колір кільця. Методи – зміна кольору кільця, зміна ширини кільця. Подія – OnClick.

3. Компонент – рівносторонній трикутник. Властивості компонента – довжина сторони, колір заповнення, стиль заповнення. Методи – поворот, зміна довжини сторони, зміна кольору заповнення Подія – OnClick.

4. Компонент – прямокутник. Властивості компонента – довжини двох сторін, колір заповнення, стиль заповнення, колір контуру. Методи – обмін сторін (swap), зміна довжин сторін, зміна кольору заповнення і кольору контуру. Подія – OnClick.

5. Компонент – еліпс. Властивості компонента – довжини півосей, колір заповнення, стиль заповнення. Методи – обмін півосей (swap), зміна довжин півосей, зміна стилю заповнення і кольору контуру. Подія – OnClick.

6. Компонент – сектор круга. Властивості компонента – радіус, початковий кут, кінцевий кут, колір заповнення, стиль заповнення. Методи – поворот, зміна кінцевого кута, зміна кольору заповнення Подія – OnClick.

7. Компонент – сегмент круга. Властивості компонента – радіус, центральний кут, початковий кут, колір заповнення, стиль заповнення. Методи – поворот, зміна

центрального кута. Подія – OnClick.

8. Компонент – спеціалізоване поле редагування і вводу TEdit для масиву цілих чисел. Властивості компонента – Text з цифрами і пробілами, розмірність, масив утворених цілих чисел, максимум масиву. Методи – сортування масиву, реверсування масиву. Подія – OnChange.

9. Компонент – спеціалізоване поле редагування і вводу TEdit для дійсних масиву чисел з дробовою крапкою. Властивості компонента – Text з послідовності дійсних чисел у символьному вигляді, розмірність, масив утворених дійсних чисел, мінімум цілої частини елементів масиву. Методи – формування рядка з цілими частинами, перевірки відсутності у чисел дробової частин. Подія – OnChange.

10. Компонент – годинник з часовою і хвилинною стрілками Властивості компонента – радіус, кути часової і хвилинною стрілок. Методи – встановлення поточного часу, зміна часу на задану кількість хвилин. Подія – OnClick.

11. Компонент – електронний годинник з показаннями годин, хвилин і секунд. Властивості компонента – формат часу для демонстрації (два типи формату), години, хвилини, секунди. Методи – встановлення поточного часу, зміна часу на задану кількість хвилин, зміна формату часу. Подія – OnClick.

12. Компонент RegExpMemo – багаторядкове поле редагування з можливістю використання пошуку за шаблоном з метасимволами. Властивості компонента – рядок з шаблоном, індекс початку для пошуку ланцюжка, колір виділення. Методи – пошук ланцюжка, що відповідає шаблону, довжина знайденого ланцюжка, зміна індексу початку пошуку. Подія – OnChange. Для реалізації задачі використати клас regexp, підключивши його до проекту директивою #include <regexp>.

4. Лабораторна робота №3 (для спеціалістів). Використання графіки

Варіанти завдань

13. Зобразити перетворення квадрата в круг. Радіус круга і швидкість перетворення задається користувачем.

14. Зобразити вільне падіння краплі на тверду поверхню. Масу краплі та її початкову швидкість задає користувач.

15. Зобразити рух кульки по еліптичній траєкторії. Надати користувачу можливість задавати швидкість кульки та ексцентриситет еліпсу.

16. Зобразити обертання колеса навколо своєї осі. Радіус колеса та швидкість обертання задаються користувачем.

17. Намалювати хвилю (синусоїду) яка рухається. Напрямок, швидкість руху та висоту хвилі має вибирати користувач.

18. Зобразити спіраль Архімеда, що обертається навколо свого центру. Напрям, швидкість обертання, а також параметри спіралі задаються користувачем.

19. Зобразити рух маятника із врахуванням сили тяжіння Землі. Маса, довжина, початкове положення і швидкість маятника задаються користувачем.

20. Зобразити зірку яка обертається навколо своєї осі симетрії. Надати можливість користувачу визначати швидкість обертання та розміри зірки.

21. Зобразити олівець якій малює синусоїду $a \times sin(b \times x)$. Надати користувачу можливість задавати значення параметрів *a* та *b*.

22. Зобразити рух кульки всередині поверхні, обмеженої колом. Радіус кола, початкова швидкість кульки, напрям, її початкове положення задаються користувачем.

23. Зобразити збільшувальне скло (коло) яке рухається над рядком тексту. Літери, які потрапляють в коло, зображати вдвічі більшими. Рядок тексту та швидкість руху кола задає користувач.

24. Зобразити куб якій обертається навколо осі OZ. Надати користувачу можливість задавати довжину ребра куба та швидкість обертання.

25. Зобразити процес накачування м‘ячика. Критичний тиск у м‘ячику і швидкість накачування задаються користувачем.

26. Зобразити рух декількох кульок всередині області, обмеженої прямокутником. Початковий напрямок руху кульок вибирається випадковим чином. Розміри прямокутника, швидкість руху кульок, а також їх кількість (від 1 до 5) задається користувачем.

27. Зобразити перетворення круга в зірку зі збереженням площі. Радіус круга і швидкість перетворення задаються користувачем.

28. Зобразити процес вимальовування графіків деяких функції функцій на одній координатній площині різними кольорами. Для вибору функцій використати TCheckBox. Для вибору кольорів застосувати вбудоване стандартне діалогове вікно.

29. Зобразити відкриття/закриття парасолі (по натисканню клавіші Enter) і поворот парасолі праворуч/ліворуч (по натисканню правої/лівої клавіш миші).

30. Розробити простий графічний редактор, який дозволяє компонувати малюнки за допомогою динамічно створених компонентів типу TShape. Дати можливість вибирати тип кожної фігури (властивість Shape), з яких складається майбутній малюнок, та її колір (групи радіокнопок).

5. Лабораторна робота №4. Доступ до баз даних

Розробити базу даних у довільному середовищі (Access, Visual FoxPro і т.і.). Занести у таблицю бази даних не менше 10 записів. Для доступу до бази даних встановити псевдонім (alias) через адміністратор джерел даних ODBC. У проекті C++Builder до створеної бази даних звертатися за її псевдонімом, використовуючи механізми доступу із застосуванням ADO + OLE DB + ODBC для непарних номерів і BDE + ODBC Link + ODBC для парних номерів.

У проекті

- створити дві форми: відокремити модуль даних та інтерфейсну частину проекту;
- створити SQL-запити, використовуючи властивість елемента керування типу TQuery або елемента керування типу TADOQuery;

- використати додаткове обчислювальне поле (Calculate) в таблицях баз даних, добавлене з середовища C++Builder;
- використати фільтри (властивості Filter і Filtered);
- створити динамічні запити з параметрами (властивість Params для компонента TTable, або властивість Parameters для компонента TADOTable).

Для реалізації вище перерахованого при необхідності доповнити умову задачі.

Варіанти завдань

1. **1.** Відомості про книги – це прізвище автора, назва, рік видання, вартість. Є база даних з таблицею, записи якої – відомості про книги. Створити запит на книги автора Іванова 2006-2009 років видання, впорядкувавши назви за алфавітом. Надрукувати інформацію про всі книги, знизивши їх вартість на 10% (обчислювальне поле).

1. **2.** Інформація про автомобіль складається з номера, марки, року випуску прізвища. Є база даних з таблицею, записи якої – інформація про автомобілі. Створити запит на автомобілі ВАЗ з номерами, більшими 8800. Надрукувати інформацію про всі автомобілі у хронологічному порядку років випуску. Добавити обчислювальне поле з номерами, в яких збільшити всі номера (без буквеної частини) на 100.

1. **3.** Відомості про учня складаються з його прізвища , імені, домашньої адреси, номера телефону. Є база даних з таблицею, записи якої – інформація про учнів. Створити запит на прізвища учнів з вулиці Головна, упорядкувавши їх за алфавітом. Створити нове обчислювальне поле в таблиці, об'єднавши адресу з телефоном. Вивести таблицю з прізвищем і новим полем.

1. **4.** У таблиці задано інформацію про назву міста, кількість інститутів, чисельність населення, чисельність студентів. Впорядкувати таблицю по спаданню кількості інститутів, створивши в ній нове обчислювальне поле: відношення кількості студентів до чисельності населення. Створити запит на назви міст з кількістю інститутів більшою за 3.

1. **5.** Є база даних з таблицею, записи якої – інформація про ліки, які зберігаються на складі: назва, термін зберігання, дата випуску, кількість одиниць, вартість. Створити запит на ліки, для яких вже вийшов термін зберігання. Вивести таблицю з назвами ліків та їх загальною вартістю (створити обчислювальне поле).

1. **6.** На складі зберігається продукція заводу. Є база даних з таблицею, записи якої – інформація про кожний вид продукції: номер продукції, назва, кількість одиниць, дата випуску, вартість одиниці. Створити запит на назви продукції, випущеної з 2007 по 2009 роки, відсортувавши їх за алфавітом. Створити нове обчислювальне поле з повною вартістю продукції. Вивести таблицю з номером продукції та новим полем.

1. **7.** Є база даних з таблицею, записи якої – інформація про точки у просторі: координати точки, назва точки, маса точки. Створити запит на записи назвами і масами точок, впорядкований по назвах. Вивести таблицю з назвами точок і відповідними відстанями від точок до початку координат (створити обчислювальне поле).

1. **8.** Є база даних з таблицею, записи якої – інформація про прямі у просторі: коефіцієнти рівняння прямої, назва прямої, колір прямої. Створити запит на записи з назвами і кольором прямих, які належать координатним площинам. Впорядкувати записи по назвах. Вивести таблицю з назвами прямих і відповідними відстанями від прямих до початку координат (створити обчислювальне поле).

1. **9.** Є база даних з таблицею, записи якої – інформація про файли: назва файлу, розширення, дата створення, розмір, атрибут. Створити запит на файли з розширенням .exe, розмір яких більший за 100 Кб. Впорядкувати записи по назвах. Вивести таблицю з назвами файлів і примітками (створити обчислювальне поле) про то, чи файл можна виконувати , чи ні (розширення .exe, .bat, .com).

10. Є телефонна база даних з таблицею, записи якої складаються з п'ятизначного номера телефона, прізвища або назви організації, адреси. Створити запит на список номерів, які починаються на 3, впорядкувавши прізвища за алфавітом. Вивести таблицю, змінивши всі номери, які починаються на 2 на такі ж, але з першою цифрою 5. Завести для цього додаткове обчислювальне поле.

11. Є база даних електромережі з таблицею, записи якої складаються з прізвища квартиронаймача, початкових і кінцевих показників лічильника за місяць, тарифу, кількості кіловат. Створити запит на прізвища квартиронаймачів, у яких кінцевий показник лічильника більший за 12345, впорядкувавши їх за алфавітом. Вивести таблицю з новим полем, в якому визначається сума до оплати (створити обчислювальне поле).

12. Є база даних по навантаженню викладача з таблицею, записи якої складаються з назви предмету, номера курсу, кількості студентів, кількості лекційних і лабораторних годин, ознаки наявності заліку або іспиту. Створити запит на предмети, які викладаються на 5 курсі і для яких є залік, впорядкувавши назви за алфавітом. Вивести таблицю, зменшивши кількість лабораторних годин на 4 (створити обчислювальне поле).

13. База даних містить таблицю з інформацією про кімнати у гуртожитку. Запис таблиці має вигляд: кількість мешканців, максимально можлива кількість мешканців, номер кімнати, площа кімнати, факультет. Створити запит про кімнати, у яких найменша площа на одного студента, згрупувавши записи за факультетами. Вивести таблицю, добавивши обчислювальне поле з кількістю незайнятих місць у кімнаті.

14. База даних містить таблицю з інформацією про квартири у домі. Запис має вигляд: прізвище власника, кількість зареєстрованих мешканців, площа, число кімнат, поверх. Вивести дані про квартири із площею менше 50 кв. м., впорядкувавши записи за кількістю кімнат.

Вивести таблицю, додавши в неї обчислювальне поле з площею, яка припадає на кожного мешканця квартири.

15. База даних містить таблицю з інформацією про поїзди. Запис має вигляд: номер потягу, пункт призначення, час відправлення, час прибуття, кількість вільних місць. Отримати довідку про наявність вільних місць на заданий номер потягу. Вивести дані про потяги із кількість вільних місць більшої 60, впорядкувавши записи за часом відправлення. Вивести таблицю, добавивши в неї обчислювальне поле з часом у дорозі.

16. База даних – відомості про птахів. У таблиці міститься інформація про назву птаха, масу, розмах крила, ціну. Вивести інформацію про всіх птахів; інформацію про птахів, розмах крил яких більше заданого. Створити обчислювальне поле – відношення маси до розмаху крила.

17. База даних – інформація про кактуси в оранжереї. У таблиці містяться поля: назва кактуса, час посадки, висота, ціна, колір, колючість, кількість. Вивести інформацію про всі кактуси. Вибрати кактуси старші 5 років. Створити обчислювальне поле – вік кактуса.

18. База даних – інформація про ботанічний сад. У таблиці містяться поля: назва дерева, час посадки, висота, товщина стовбура. Вивести інформацію про дерева. Вибрати дерева з товщиною стовбура більшою за 15см. Додати обчислювальне поле – вік дерева.

19. База даних – інформація про телевізори. У таблиці містяться поля: марка, фірма-виробник, дата випуску, гарантійний термін, ціна, кількість одиниць на складі. Вибрати телевізори з ціною більшою за 100$. Додати обчислювальне поле – час, який залишився до кінця гарантійного терміну.

20. База даних – інформація про канцелярські приладдя. У таблиці містяться поля: назва, кількість, вартість, виробник, постачальник. Вибрати канцелярські приладдя з ціною більшою за 10 грн. Додати обчислювальне поле – вартість товарів певної назви.

21. Є база даних спортивних шкіл (номер, адреса, кількість учнів, рік заснування). Створити запит „кількість учнів у спортивних школах” (номер школи, кількість учнів), впорядкувавши кількість учнів за спаданням. Відобразити п‘ять найстаріших шкіл.

22. Є база даних „Вчитель”. Таблиця містить інформацію про учнів: порядковий номер, ПІП, оцінки за предмети (математика, фізика, хімія, біологія, історія). Створити запит, що виведе інформацію про успішність учнів по математиці (ПІП, оцінка, впорядкована за спаданням). Засобами C++Builder створити обчислювальне поле – середня оцінка (№, ПІП, середня оцінка).

23. Є база даних „Нерухомість”. Таблиця містить інформацію про доступну нерухомість: тип, адреса, площа, ціна. Створити запит – 5 найдорожчих пропозицій. Засобами C++Builder створити обчислювальне поле – середня ціна 1 кв.м. (№, ПІП, середня оцінка).

24. Є база даних „Комп‘ютери”. Таблиця містить інформацію: назва, тип, кількість процесорів, об‘єм оперативної пам‘яті, об‘єм вінчестера, наявність CD-ROM, рік випуску, ціна. Створити запит, що виведе інформацію про вік комп‘ютерів. Засобами C++Builder створити обчислювальне поле, що характеризує співвідношення об‘єму оперативної пам‘яті до об‘єму вінчестера .

25. Є база даних „Користувачі ПК”. Таблиця містить інформацію: логін, пароль, ПІП користувача, дата реєстрації, дата останнього входу в систему. Вивести інформацію про 5 найактивніших користувачів.

26. Є база даних „Книги”. Таблиця містить інформацію: назва, автор, рік видання, дата поступлення в магазин, ціна за одиницю, кількість на складі. Створити запит, що виведе інформацію про 10 найстаріших книг (за роком видання). Засобами C++Builder створити обчислювальне поле, що характеризує сумарну вартість книг для кожної назви.

27. Є база даних „Веб-сайти”. Таблиця містить інформацію: назва сайту, веб-адреса, кількість відвідувачів у день. Створити запит, що виведе інформацію про 10 найпопулярніших сайтів. Добавити обчислювальне поле ”кількість символів у назві сайту”.

28. Є база даних „Картини”. Таблиця містить інформацію: назва картини, дата написання, автор, дата поступлення, вартість. Вивести 10 найстаріших картин і їхню вартість. Добавити обчислювальне поле ”час знаходження у магазині”.

29. Є база даних „Туристичні маршрути”. Таблиця містить інформацію: назва маршруту, кількість місць, вартість, кількість днів, наявність знижок для постійних клієнтів, можливість сімейного відпочинку. Створити запит на туристичні маршрути із знижками без можливості сімейного відпочинку. Добавити обчислювальне поле ”вартість одного дня у маршруті”.

30. Є база даних „Транспортні маршрути”. Таблиця містить інформацію: маршрут, початкова станція, кінцева станція, час відправлення, час прибуття, вартість квитка. Добавити обчислювальне поле ”час у дорозі”. Вивести 5 найтриваліших маршрутів та їхню вартість.