23 варіант

1. Локальні та глобальні змінні.

Частини програми, звертання до яких дозволено з будь-якої її точки – змінні з глобальною видимістю. Такими, наприклад, є всі функції, що входять до складу програми, а також змінні й структури даних, визначені поза певною функцією. Навпаки, ті змінні, область доступу до яких обмежена деякою частиною вихідної програми, ми будемо називати змінними з локальною видимістю.

2. Класи пам'яті. Перевантаження функцій.

визначає її час життя, протягом якого ця змінна існує у пам'яті. Одні змінні існують недовго, інші неодноразово створюються і знищуються, треті існують протягом виконання програми. У C++ Builder є чотири специфікації класу пам'яті: auto, register, extern і static.

Ключові слова *auto* та *register* використовуються для оголошення змінних з локальним часом життя. Ці специфікації можна застосовувати лише до локальних змінних. Локальні змінні створюються при вході до блоку, в якому вони оголошені, існують лише під час активності блоку і зникають при виході з блоку.

Ключові слова *extern* і *static* використовуються, щоб оголосити ідентифікатори змінних як ідентифікатори статичного пам'яті з глобальним часом життя. Такі змінні існують з початку виконання програми. Для таких змінних пам'ять виділяється та ініціалізується одразу після початку виконання програми.

Перевантаження функцій — це можливість визначати декілька функцій з одним і тим же ім'ям, але з різними параметрами.

```
1 int subtract(int a, int b); // цілочисельна версія
2 double subtract(double a, double b); // версія типу з плаваючою крапкою
```

При виклику, компілятор визначить сам, яку версію *subtract()* слід викликати на основі аргументів, які використовуються у виклику функції. Якщо параметрами будуть змінні типу int, то мова C++ розуміє, що ми хочемо викликати *subtract(int, int)*. Якщо ж ми надамо два значення типу з плаваючою комою, то мова C++ зрозуміє, що ми хочемо викликати *subtract(double, double).*

3. Написати функцію, яка порівнює два цілих числа та повертає результат порівняння у вигляді одного зі знаків <, >,=

```cpp
#include <iostream>

char bilOrmen(int a, int b) {
    if(a > b) return '>';
    else if(a < b) return '<';
    else return '=';
}

int main()
{
    std::cout << bilOrmen(2, 4) << std::endl ;
    std::cout << bilOrmen(4, 4) ;

    return 0;
}
```

```
OUTPUT
[Running] cd "c:\projects\helloworld\" && g++ lab.cpp
<
=
```