1. Що таке функція?

Функція (function) - це підпрограма, яка може отримати дані та повертати деяке значення

1. Що таке заголовок функції?

Визначення функції (function definition) складається з заголовка функції і тіла.

|  |
| --- |
| **int sum(int a, int b)**  {  return a + b;  } |

1. Що таке тіло функції?

тіло функції - блок з програмним кодом, який виконується під час виклику функції.

1. Що таке прототип функції?

Оголошення функції без визначення. Прототип функції складається з типу результату функції, імені та списку параметрів. Список параметрів (parameter list) - це список всіх аргументів та їх типів, розділених комами.

1. Чи є імена формальних параметрів завжди обов'язковими в заголовку функції?

У випадку, коли параметри відсутні, дужки залишаються порожніми або містять ключове слово (void). Формальні параметри функції локалізовані в ній і недоступні для будь-яких інших функцій.

Список формальних аргументів має такий вигляд:

([const] тип 1 [параметр 1], [const] тип 2 [параметр 2], . . .)

1. Як створити локальну функцію?

Функцію не можна визначити всередині іншої функції. У C++ немає локальних функцій.

1. Як здійснюється передача параметрів у функцію?

Аргументи передаються у функцію в тому порядку, в якому вони були оголошені і визначені. Будь-який вираз C++ може бути аргументом функції, в тому числі константа, математичні та логічні вирази, виклик інших функцій, які повертають значення.

1. У чому різниця між формальними і фактичними параметрами?

Аргументи, які передаються під час виклику функції мають назву фактичних параметрів. Викликана функція має доступ до переданої інформації через так звані формальні параметри.

1. Чи є інструкція return завжди обов'язковою?

Функції усіх типів (за винятком void) повинні містити інструкцію return з таким виразом. Інструкція return завжди завершує виконання функції. Результат функції main() може бути використаний операційною системою як код помилки (0 - помилок немає)

1. Як використовувати результат функції з типом результату void?
2. Що таке локальна змінна?

Аргументи, передані у функцію, є локальними для функції. Зміни значень аргументів всередині викликаної функції не впливають на значення в функції, з якої було здійснено виклик. Це відомо як передача за значенням і означає, що у функції створена локальна копія кожного аргументу.

Змінні можна створити всередині тіла функції, або іншого програмного блоку. Такі змінні є локальними змінними

1. Що таке область видимості (scope)?

визначає, де саме в програмі цей об'єкт може бути доступний.

1. Що таке глобальна область видимості?

Глобальні змінні мають глобальну область видимості. Вони доступні в будь-якому місці програми. Глобальні змінні, визначені поза тілом функції, доступні з будь-якої функції в програмі, у тому числі з функції main()

1. Як отримати доступ до імен з глобальної області видимості?

Локальні змінні з тим же ім'ям, що й глобальні змінні, не пов'язані з відповідними глобальними змінними. Але локальна змінна з таким же ім'ям, як і глобальна змінна, приховує глобальну змінну. В таких випадках дозволу області оператор (: :) для доступу до глобальної змінної використовують операцію доступу до області видимості :: (scope resolution operator). Наприклад:

int k = 1;

void f()

{

int k = 2;

cout << k; // 2

cout << ::k; // 1

}

1. Чи можуть локальні змінні приховати глобальні змінні?

Локальні змінні з тим же ім'ям, що й глобальні змінні, не пов'язані з відповідними глобальними змінними. Але локальна змінна з таким же ім'ям, як і глобальна змінна, приховує глобальну змінну. В таких випадках дозволу області оператор (: :) для доступу до глобальної змінної використовують операцію доступу до області видимості :: (scope resolution operator). Наприклад:

1. У чому різниця між статичними і нестатичними локальними змінними?

Як правило, локальні змінні, створені у функції, зникають після виходу з функції. Коли виклик функції здійснюється знову, зміні створюються та ініціалізуються заново. Якщо треба, щоб дані зберігалися протягом усього життєвого циклу програми, така змінна може бути визначена як статична (з модифікатором static). Ініціалізація виконується тільки тоді, коли функція викликається вперше, а змінні зберігають свої значення проміж викликами функції.

1. Яким є життєвий цикл статичної локальної змінної?

протягом усього життєвого циклу програми

1. Як використовують статичні локальні змінні?

Тільки у межах функції

1. Що таке рекурсія?

Функція може викликати себе. Такий механізм має назву рекурсії.

Безпосередня рекурсія передбачає виклик функції безпосередньо з цієї функції, опосередкована передбачає, що перша функція викликає другу, а потім друга викликає першу функцію.

1. Що таке функція-підстановка (inline)?

Якщо функція оголошена як вбудована, компілятор підставляє в точку виклику її тіло. Доцільно з модифікатором inline описувати найпростіші функції, багаторазова підстановка яких не істотно вплине на розміри програми.

1. Як перевантажити ім'я функції?

Мова C++ дозволяє створити декілька функцій з однаковими іменами. Це називається перевантаженням імен функцій

1. Чи можна створити дві глобальні функції з однаковими іменами і списками параметрів, але з різними типами результату?

Примітка: дві функції з однаковим ім'ям і списком параметрів, але з різними типами результату, генерують помилку компіляції:

|  |
| --- |
| int f(double x);  double f(double x); // Помилка! |

1. Як описати аргументи з усталеними значеннями?

У багатьох випадках для аргументів функції можна вказати усталені значення (default values), які можуть бути цілком задовільними в багатьох випадках. Для вирішення цієї проблеми реалізовано механізм аргументів з усталеними значеннями, що дозволяє визначати лише ті аргументи функції, які є істотними в конкретному контексті. Наприклад:

|  |
| --- |
| int sum(int x, int y = 0, int z = 0)  {  return x + y + z;  }  int main()  {  cout << sum(5) << endl; // 5, y = 0, z = 0  cout << sum(1, 2) << endl; // 3, z = 0  cout << sum(1, 2, 5) << endl; // 8  return 0;  } |

* Слід пам'ятати, що аргументи з усталеними значеннями повинні бути останніми в списку
* Усталене значення не може бути повторене в більш пізніх описах функції, навіть якщо перевизначення ідентичне оригіналу. Таким чином, наведений нижче код видає помилку:

1. Що таке посилання?

Посилання (reference) може бути визначене як друге ім'я (псевдонім) існуючої змінної (об'єкта).

Можна описати посилання зазначенням типу, за яким слідує оператор посилання (&) та ім'я посилання:.

|  |
| --- |
| int i = 10;  int &j = i; // посилання на i  j = 11;  cout << i; // 11 |

1. Як ініціалізувати посилання?

Не можна створювати посилання, на вказуючи об'єкт, на який ми посилаємось. Посилання повинні бути ініціалізовані під час створення:

int &k; // синтаксична помилка!

Можна визначити посилання на константний об'єкт:

|  |
| --- |
| int m = 2;  const int &n = m; // посилання на i  double x = n + 1; // OK  m = 11; // OK  n = 12; // помилка! |

1. Для чого використовують посилання?

* Посилання можуть бути використані як аргументи функції. Наприклад, можна створити функцію, яка обмінює значення двох цілих змінних:
* Посилання можуть бути використані як тип результату функції. Дуже важливо, щоб гарантувати, що посилання пов'язане зі змінною, яка існує після виходу з функції:

1. Чи можна повернути посилання з функції?
2. Чи можна повертати посилання на нестатичну локальну змінну?