Структура ENUM в мові програмування C

Кучеренко Володимир

24 листопада 2023 р.

Вступ

Вітаю! Ви переглядаєте курсову роботу Кучеренка Володимира, студента 2-го курсу комп'ютерної математики. Цей текстовий файл описує структуру ENUM та її використання в мові програмування C.

Структура ENUM та історія

enum (перерахування) в мові програмування C є структурою даних, яка дозволяє створювати множину іменованих цілих констант, які можна використовувати у програмі. Вони полегшують читання та розуміння коду, замінюючи "магічні числа"іменами з сенсом.

Історія *enum* пов'язана з виникненням мови C, що спростила роботу з константами та полегшила розуміння коду.

Основні особливості та використання *enum*

[label=–]**Іменовані константи:**
enum Days {MON, TUE, WED, THU, FRI, SAT, SUN}; У цьому прикладі *MON*, *TUE*, і так далі - це іменовані константи, що представляють дні тижня. **Значення за замовчуванням:**
За замовчуванням, перша константа отримує значення 0, а кожна наступна - на одиницю більше. **Явне визначення значень:**
enum Months {JAN = 1, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC}; Ви можете явно вказати значення для констант, коли це необхідно. **Використання для уникнення "магічних чисел":**
enum State {START, PROCESSING, ERROR, FINISHED} Замість використання чисел прямо в коді, використання *enum* робить код зрозумілим і підтримуваним. **Варіанти для станів та типів:**
enum Color {RED, GREEN, BLUE}

Загалом, *enum* є корисним інструментом для полегшення зрозуміння та обслуговування коду, особливо при роботі з константами, що мають конкретні значення або визначають різні стани програми.

Використання ENUM в C++

В мові програмування C++, перерахування (*enum*) використовується для визначення нового типу даних, який складається з констант, заздалегідь визначених значень. У C++ є деякі додаткові можливості порівняно з C, які полегшують роботу з перерахуваннями.

Типи перерахувань в C++

У C++ є два способи визначення перерахувань: *enum* і *enum class* (який ще називається "strongly typed enum"або "scoped enum"). Можна просто порівняти їх основні відмінності:

[label=–]**Область видимості:**

- *enum*: Константи мають глобальну область видимості і можуть конфліктувати з іншими іменами в програмі.
- *enum class*: Константи входять в область видимості самого перерахування, тому їхнє використання вимагає приставки перерахування.

- **Типи та безпека:**

  - *enum*: Має неявні цілочисельні значення. Можливі неочікувані порівняння та конвертації.
  - *enum class*: Забороняє неявні конвертації і порівняння з цілими числами. Значення в межах *enum class* не конфліктують з іншими типами.

- **Порівняння:**
  - *enum*: Дозволяє порівнювати з іншими цілочисельними типами без явного приведення.
  - *enum class*: Вимагає явного приведення для порівняння з іншими типами.

- **Оператори приведення:**
  - *enum*: Дозволяє неявне приведення до цілочисельних типів.
  - *enum class*: Вимагає явного приведення до інших типів.

- **Кількість констант з однаковим ім'ям:**
  - *enum*: Дозволяє кілька констант з однаковим ім'ям в різних перерахуваннях.
  - *enum class*: Імена констант унікальні в межах свого перерахування.

- **Автоматичне приведення до цілих чисел:**
  - *enum*: Автоматичне приведення до цілочисельних типів дозволяє використання у виразах без явного приведення.
  - *enum class*: Не має автоматичного приведення, що підвищує безпеку, але вимагає явного приведення до цілих чисел.

Загалом, *enum class* надає більше контролю та безпеки при використанні перерахувань в C++.

Підсумок

На цьому завершується моя робота. Дякую за увагу та час, що ви відвели на читання. Ставте вподобайки та високі оцінки. Бажаю успіхів у вашій навчальній діяльності та до нових зустрічей!