Звіт

Лабораторна робота №5

студента групи ТТП-32

Ходакова Максима

**Варіант 10**

**Умова**

Запрограмувати взаємодію 2 потоків, що реалізують деякі функції int f(int x) {...} та int g(int x) {...}  
Вміст функцій f і g треба буде змінити при здачі перед запуском на вимогу викладача.  
Врахувати, що функція може повертати:  
1) 0 (трактується як false для логічних операцій)  
2) інше ціле число (трактується як true для логічних операцій)  
3) **нічого не повертати, зациклюватись** (можна змоделювати нескінченим циклом while(1); або sleep(10...0); )  
Тобто функції f та g можуть бути **частково визначені** (тобто «зациклюватись» і ніколи не повертати результат). Потрібно *коректно опрацювати* таку ситуацію і *запитати користувача*: «1) продовжити обчислення, 2) припинити або 3) продовжити, не перепитуючи більше» наприклад, кожні 10 секунд.  
  
Також врахувати, що функції f(x) і g(x) є "**чистими функціями**" (<https://en.wikipedia.org/wiki/Pure_function>), тобто вони займаються тільки обчисленням значення над вхідним аргументом, вони **не обробляють ніяких інших запитів** (у тому числі – про завершення обчислень) і **не взаємодіють з іншими процесами та потоками** ні в який інший спосіб, окрім викликів обчислень f(x) і g(x) (тобто запуску функції на обчислення) та повернення результату (коли обчислення результату завершено) через return(...);.  
  
Потрібно обчислити деякий результат над f і g (**згідно варіанту нижче, за номером у списку групи**), враховуючи, що f і g можуть нічого не повернути (зависнути, невизначеність), а також спираючись на наступні **правила обчислень (логіка Кліні)** (<https://en.wikipedia.org/wiki/Three-valued_logic#Kleene_and_Priest_logics>):  
1) x && false == false && x == false  
2) x || true == true || x == true  
3) 0 \* x == x \* 0 == 0 для довільного числа x  
Основна ідея – "**ліниві**" симетричні (комутативні) **обчислення**: зупинятись і видавати результат, як тільки є найменші підстави це зробити, тобто як тільки з якихось компонентів обчислення стає зрозуміло, що результат вже визначено і він надалі не зміниться (бо не залежить від недообчислених компонентів виразу).

10. **Взаємодія процесів. Паралелізм. Обмін повідомленнями**. Обчислити f(x) \* g(x), використовуючи 2 допоміжні процеси: один обчислює f(x), а інший – g(x). Основна програма виконує ввод-вивід та операцію \*. Використати обмін повідомленнями між процесами (Messages). Реалізувати варіант **блокуючих** операцій обміну повідомленнями, тобто з очікуванням обробки повідомлення і відповіді на повідомлення (і “зависанням” процесу на цей час). Функції f(x) та g(x) “нічого не знають друг про друга” і не можуть комунікувати між собою.