Lập 1 chương trình trên ngôn ngữ lập trình tùy chọn C++/Java/C# gồm 3 tác vụ (Task1, Task2, Task3) hoạt động ở chế độ đồng bộ và không đồng bộ như sau:

Task1:

1. (2 điểm) Viết 1 hàm gồm 1 vòng lặp nhập liên tục các kí tự từ bàn phím, lưu các kí tự nhập vào một biến tổng thể C có cấu trúc dữ liệu kiểu bộ đệm vòng, và vòng lặp dừng lại khi nhập kí tự ‘!’. Viết hàm main() gọi hàm, dịch và chạy chương trình.

2. (3 điểm) Thiết lập Task1 bằng cách đặt hàm trên vào 1 luồng, kích hoạt luồng trong hàm main(). Dịch và chạy lại chương trình.

Task2:

3. (2 điểm) Viết 1 hàm có 1 vòng lặp chỉ dừng lặp khi vòng lặp của Task1 dừng lại. Vòng lặp này luôn hiển thị kí tự theo thứ tự nhập được lưu trong biến tổng thể C của Task1. Bổ sung lệnh gọi hàm trong hàm main(), dịch và chạy lại chương trình.

4. (2 điểm) Thiết lập Task2 bằng cách đặt hàm trên vào 1 luồng, kích hoạt luồng trong hàm main(). Dịch và chạy lại chương trình.

Task3:

5. (1 điểm) Viết 1 hàm đếm số kí tự bằng ‘A’ do Task1 tạo ra.

Task3 được gọi đồng bộ với Task1 nghĩa là Task3 chỉ thực sự hoạt động khi Task1 kết thúc. Gọi Task3 trong hàm main(). Dịch và chạy lại chương trình.

Chú ý: Có thể sử dụng cấu trúc dữ liệu tương đương để thay thế kiểu cấu trúc bộ đệm vòng.

C# code này sửa lại là đáp ứng được.

Method1 trả lại con đếm và truyền tham số cho Method3

Sử dụng await trước khi truyền tham số cho Method3 và phải sử dụng từ khóa async để gọi hàm.

class Program

{

static void Main(string[] args)

{

callMethod();

Console.ReadKey();

}

public static async void callMethod()

{

Task<int> task = Method1();

Method2();

int count = await task;

Method3(count);

}

public static async Task<int> Method1()

{

int count = 0;

await Task.Run(() =>

{

for (int i = 0; i < 100; i++)

{

Console.WriteLine(" Method 1");

count += 1;

}

});

return count;

}

public static void Method2()

{

for (int i = 0; i < 25; i++)

{

Console.WriteLine(" Method 2");

}

}

public static void Method3(int count)

{

Console.WriteLine("Total count is " + count);

}

}

C# code này sửa lại là đáp ứng được.

Method1 trả lại con đếm và truyền tham số cho Method3

Sử dụng await trước khi truyền tham số cho Method3 và phải sử dụng từ khóa async để gọi hàm.

class Program

{

static void Main(string[] args)

{

callMethod();

Console.ReadKey();

}

public static async void callMethod()

{

Task<int> task = Method1();

Method2();

int count = await task;

Method3(count);

}

public static async Task<int> Method1()

{

int count = 0;

await Task.Run(() =>

{

for (int i = 0; i < 100; i++)

{

Console.WriteLine(" Method 1");

count += 1;

}

});

return count;

}

public static void Method2()

{

for (int i = 0; i < 25; i++)

{

Console.WriteLine(" Method 2");

}

}

public static void Method3(int count)

{

Console.WriteLine("Total count is " + count);

}

}

4 đề tiếp theo:

Dạng threads:

Lập 1 chương trình trên ngôn ngữ lập trình tùy chọn C++/Java/C# tạo ra 2 luồng (Thread1, Thread2) hoạt động như sau:

Thread1:

1. (2 điểm) Viết 1 hàm gồm mở 1 file chẳng hạn vd.txt ở chế độ text (giả định đây là một file có kích thước rất lớn). Hàm có 1 vòng lặp đọc từng kí tự của file và hiển thị dạng mã hexa của kí tự, vòng lặp dừng khi không đọc tiếp được (đã đọc hết).

2. (3 điểm) Đặt hàm trên vào 1 luồng, kích hoạt luồng trong hàm main().

Thread 2:

3. (2 điểm) Viết 1 hàm có 1 vòng lặp chỉ dừng lặp khi vòng lặp của Thread1 dừng lại. Vòng lặp này luôn hiển thị các kí tự đã đọc được từ file ở Thread1. Bổ sung lệnh gọi hàm trong hàm main().

4. (3 điểm) Thiết lập Thread2 bằng cách đặt hàm trên vào 1 luồng, sửa hàm main() để kích hoạt luồng Thread2 trước, rồi kích hoạt luồng Thread1. Dịch và chạy chương trình.

Dạng timer:

Lập 1 chương trình trên ngôn ngữ lập trình tùy chọn C++/Java/C# gồm 2 tác vụ (Task1, Task2) hoạt động ở chế độ không đồng bộ như sau:

Task1:

1. (2 điểm) Viết 1 hàm gồm 1 vòng lặp nhập 1 xâu kí tự st từ bàn phím (st là một biến tổng thể). Nếu xâu st được nhập là “quit” thì thoát khỏi vòng lặp, trái lại ghi vào thành một dòng text của một file text là vidu.txt. Viết hàm main() gọi hàm, dịch chương trình.

2. (3 điểm) Thiết lập Task1 bằng cách đặt hàm trên vào 1 timer hoặc 1 luồng, kích hoạt timer/luồng trong hàm main(). Dịch chương trình.

Task2:

3. (2 điểm) Viết 1 hàm có 1 vòng lặp vô hạn đọc xâu kí tự ở file vidu.txt. Đọc được 1 dòng thì đóng file. Nếu dòng đọc được là “quit” thì thoát vòng lặp, trái lại hiển thị biến xâu kí tự. Hiển thi số lượng dòng text của file vidu.txt. Bổ sung lệnh gọi hàm trong hàm main.

4. (3 điểm) Thiết lập Task2 bằng cách đặt hàm trên vào 1 timer hoặc 1 luồng, kích hoạt time/luồng trong hàm main(). Dịch và chạy lại chương trình.

Dạng lời gọi không đồng bộ:

Lập 1 chương trình trên ngôn ngữ lập trình tùy chọn C++/Java/C# gồm 2 tác vụ (Task1, Task2) hoạt động ở chế độ không đồng bộ như sau:

Task1:

1. (2 điểm) Viết 1 hàm gồm 1 vòng lặp nhập liên tục các số nguyên từ bàn phím, lưu các số nhập được vào một biến cấu trúc A kiểu bộ đệm vòng, và vòng lặp dừng lại khi nhập số nguyên < 0.

2. (3 điểm) Thiết lập Task1 bằng cách đặt hàm trên vào 1 luồng, kích hoạt luồng trong hàm main().

Task2:

3. (2 điểm) Viết 1 hàm có 1 vòng lặp chỉ dừng lặp khi vòng lặp của Task1 dừng lại. Vòng lặp này luôn hiển thị các số nguyên được lưu trong biến A. Bổ sung lệnh gọi hàm trong hàm main().

4. (3 điểm) Thiết lập Task2 bằng cách đặt hàm trên vào 1 luồng, kích hoạt luồng trong hàm main(). Hiển thị trung bình cộng các phần tử của mảng. Dịch và chạy chương trình.

Dạng truyền dữ liệu sử dụng socket

Lập 2 chương trình trên ngôn ngữ lập trình tùy chọn C++/Java/C# gồm CT1(client), CT2 (server) sử dụng Socket API, giao thức TCPIP truyền nhận dữ liệu tại IP="127.0.0.1" (localhost), port=5000 và hoạt động như sau:

CT1:

1. (2 điểm) Viết 1 hàm gồm 1 vòng lặp nhập vào từ bàn phím mảng số nguyên (nhập số phần tử n>=3, sau đó nhập từng phần tử mảng), nếu nhập n thỏa n < 3 thì thoát khỏi vòng lặp. Viết hàm main() gọi hàm.

2. (3 điểm) Bổ sung lệnh truyền giá trị n và truyền biến mảng đến cho CT2 trong vòng lặp. Dịch chương trình CT1 (không chạy chương trình).

CT2:

3. (4 điểm) Viết 1 hàm có 1 vòng lặp vô hạn nhận giá trị biến nguyên n được gửi đến từ CT1, Nếu nhận được n thì:

Trường hợp 1: Nếu n < 3 thì thoát khỏi vòng lặp.

Trường hợp 2: Trái lại, nhận tiếp mảng n số nguyên được gửi đến từ CT1 và hiển thị 3 giá trị đầu tiên xếp theo thứ tự từ cao xuống thấp của mảng sau khi nhận được (3 giá trị đầu tiên này có thể luôn thay đổi khi CT1 đang chạy). Viết hàm main(), dịch chương trình CT2.

4. (1 điểm) Đặt hàm ở câu 3 vào 1 timer, bổ sung lệnh kích hoạt timer trong hàm main(). Dịch chương trình.

Biên dịch và chạy chương trình:

+ Đóng tất cả CT1, CT2 đang chạy (nếu có).

+ Chạy chương trình CT2 sau đó chạy CT1 để xem hiệu ứng kết nối, truyền và nhận dữ liệu.