HƯỚNG DẪN GỠ LỖI (DEBUG) CHƯƠNG TRÌNH ĐƠN GIẢN

1 Giới thiệu

1.1 Debugging là gì?

Debugging trong lập trình là một quá trình tìm và giải quyết các lỗi (bugs, errors) có trong chương trình. Debugging là một kỹ thuật cần thiết, giúp lập trình viên có thể tìm ra các lỗi khó quan sát và nhận biết. Từ đó hỗ trợ cho quá trình chỉnh sửa và giải quyết lỗi.

1.2 Các kỹ thuật debugging cơ bản

Có nhiều kỹ thuật trong debugging, trong đó có 2 kỹ thuật cơ bản là:

- **Interactive debugging** - sử dụng **công cụ debugger (debugger tool)** đến từ các IDE để tương tác với mã nguồn của chương trình. Các công cụ debug này cho phép *thực thi mã nguồn từng bước* cũng như có thể *dừng chương trình tại một bước nhất định để quan sát giá trị các biến*. Từ đó, giúp lập trình viên hiểu hơn về chương trình và gỡ lỗi.
- **Print debugging** - sử dụng các câu lệnh in (`printf`/`cout` trong ngôn ngữ C/C++) để in ra màn hình luồng chạy chương trình hoặc giá trị biến.

Bài hướng dẫn này tập trung giới thiệu kỹ thuật **interactive debugging (debugging tương tác)**.

2 Hướng dẫn

2.1 Tạo project

Sinh viên tạo một project trong Visual Studio:

- Đặt tên project là **SimpleDebugPro**
- Đặt tên solution là **SimpleDebugSol**
- Tạo tập tin mã nguồn là **SimpleDebug.cpp** có nội dung như sau:

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, c;

    a = 1234567890;
    b = a * 10;
    c = b / 100;

    cout << "Gia tri c: " << c << endl;
    return 0;
}
```

Khi chạy chương trình trên, kết quả mong muốn xuất ra màn hình là:

```
Gia tri c: 123456789
```

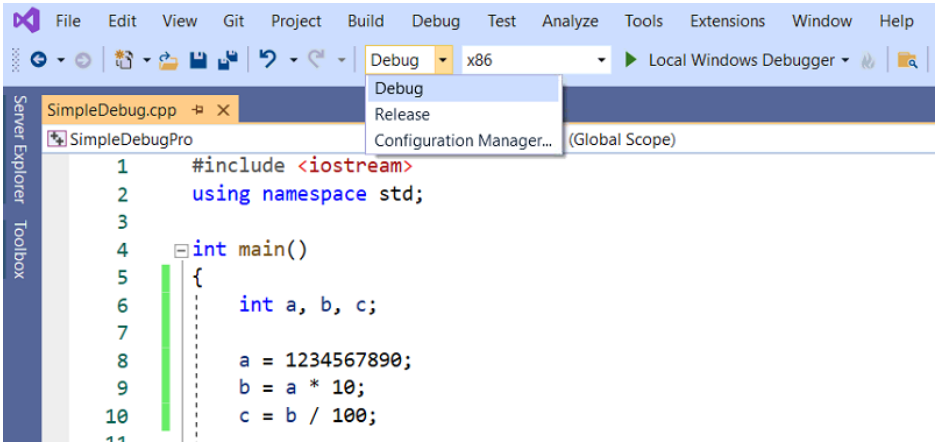
Tuy nhiên, kết quả thực tế là:

```
Gia tri c: -5392229
```

Để tìm ra giải thích cho chương trình trên, sinh viên có thể ứng dụng kỹ thuật Debugging như hướng dẫn ở các bước sau.

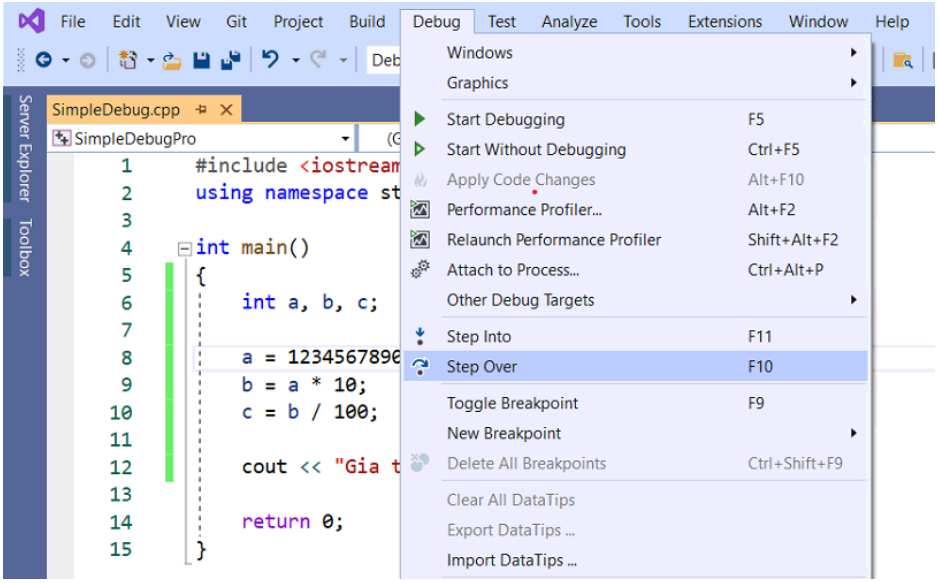
2.2 Chọn chế độ Build project

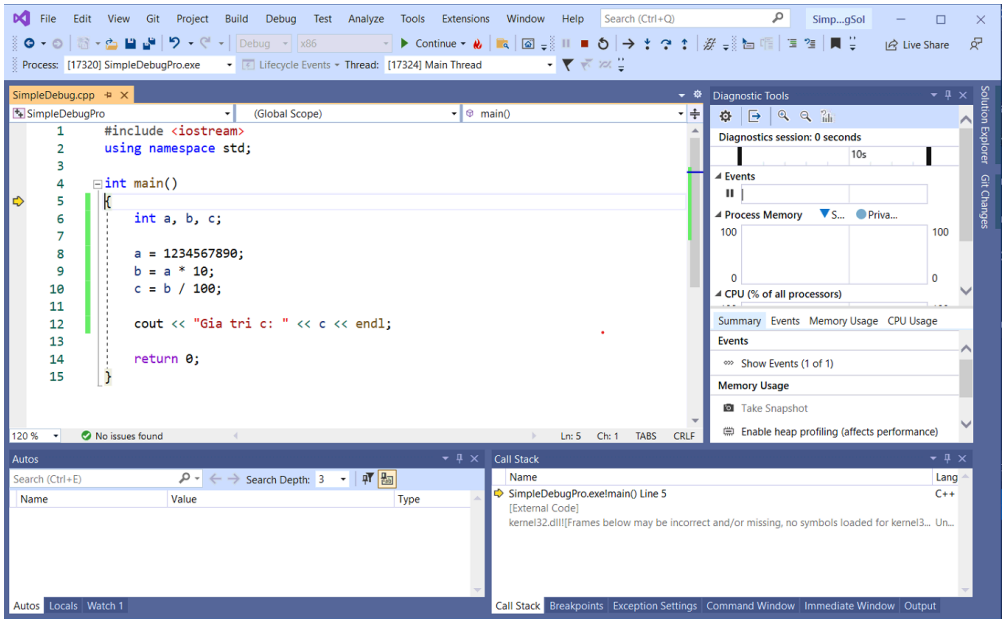
Trong Visual Studio, có 2 chế độ build project là **Debug** và **Release**. Sinh viên chọn chế độ **Debug** để debug chương trình.



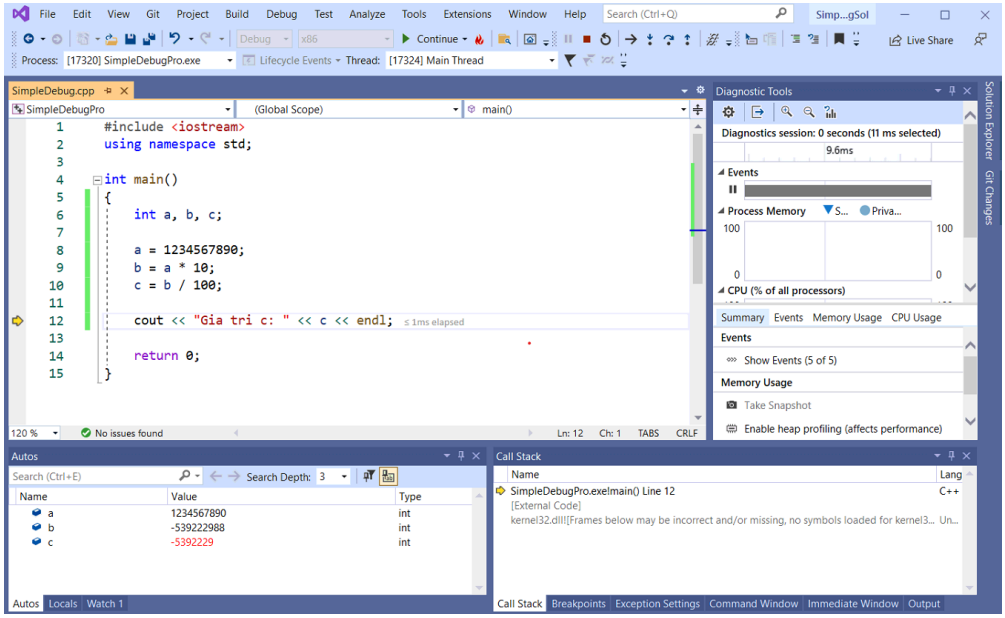
2.3 Chạy chương trình từng lệnh

Sinh viên chọn **Debug** ở thanh công cụ → **Step Over (F10)** để bắt đầu chạy chương trình từng lệnh.





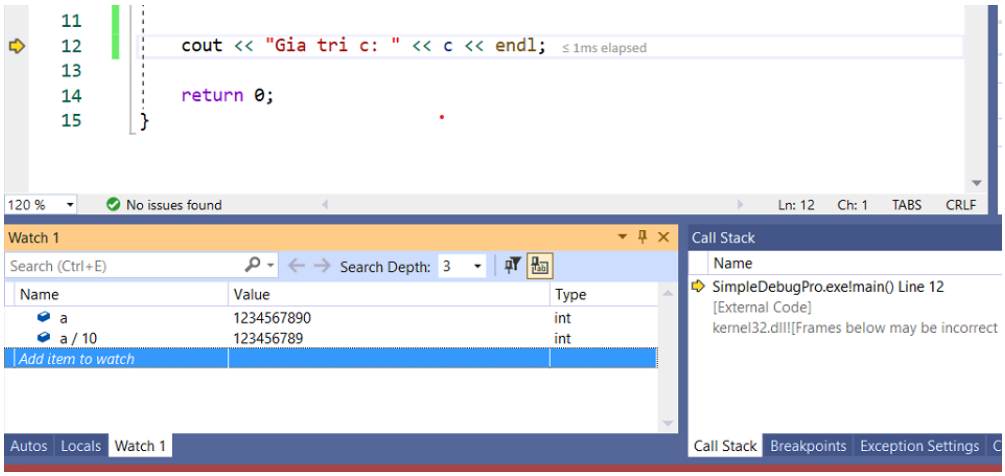
Sinh viên nhấn F10 để lần lượt đi qua từng câu lệnh. Sinh viên xem giá trị của từng biến ở **cửa sổ Autos** hoặc rê chuột vào từng biến.



Tại đây, ta có thể quan sát thấy giá trị của biến `b` KHÔNG PHẢI là `a * 10 = 12345678900`, thay vào đó là `b = -539222988` (vì `b` bị tràn số). Từ đó, ảnh hưởng đến giá trị của biến `c`. Từ nguyên nhân là `b` bị tràn số, ta có thể giải quyết bằng cách khai báo biến `b` với kiểu dữ liệu có miền giá trị lớn hơn (ví dụ `long long`).

Sinh viên cũng có thể sử dụng **cửa sổ Watch** để xem giá trị của các biến/biểu thức khác:

- **Debug → Window → Watch**...
- Nhập tên biến/biểu thức trên từng dòng của **cửa sổ Watch**.



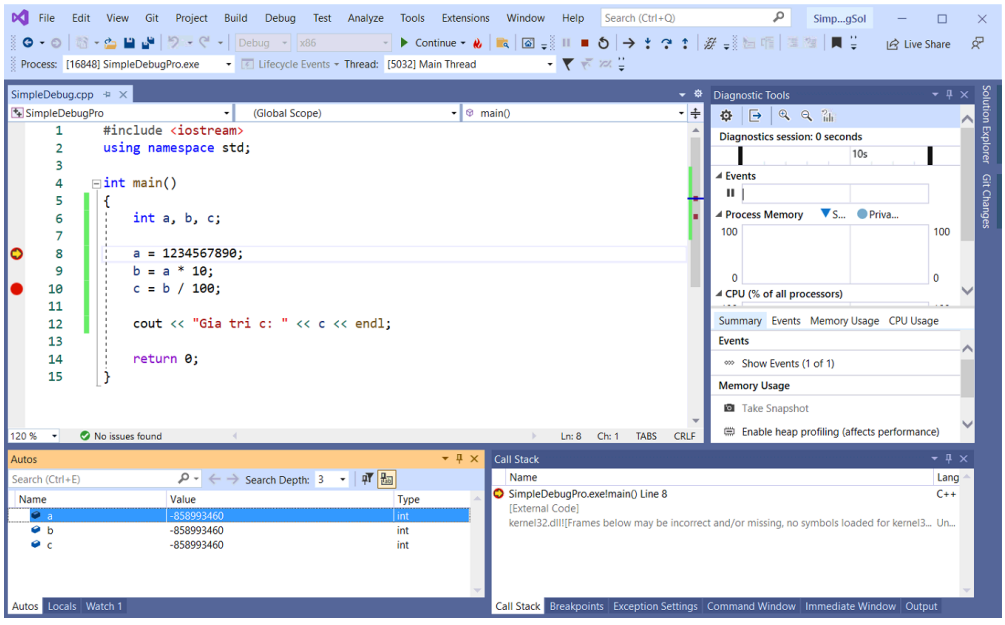
2.4 Kết thúc debugging

Để kết thúc quá trình debug, sinh viên chọn **Debug → Stop Debugging (Shift + F5)**.

2.5 Đặt Breakpoints

Để bắt đầu debug tại một dòng lệnh nhất định, sinh viên có thể sử dụng **Breakpoints**:

- Chọn dòng lệnh muốn đặt breakpoint.
- Chọn **Debug → Toggle Breakpoint (F9)**.
- Sau đó, chạy chương trình ở chế độ debugging: **Debug → Start Debugging (F5)**.
- Nhấn *F5 để đến breakpoint kế tiếp*, hoặc nhấn *F10 để chạy từng lệnh.*



3 Thực hành

Sinh viên **tạo một project Score** với mã nguồn chương trình như sau:

```
#include <iostream>
using namespace std;

int main()
{
    float diem;
    cout << "Nhap vao diem cua sinh vien: ";
    cin >> diem;

    if (diem >= 5)
        cout << "Dau" << endl;
    if (diem >= 4.5)
        cout << "Thi lai" << endl;
    if (diem < 4.5)
        cout << "Hoc lai" << endl;

    return 0;
}
```

Sau đó, sử dụng kỹ thuật **interactive debugging** để tìm và sửa lỗi chương trình trên.

4 Tài liệu tham khảo khác

Sinh viên có thể tham khảo thêm về hướng dẫn debug trên Visual Studio tại Tutorial: Learn to debug C++ code using Visual Studio.

HẾT