Lesson 5:

Syntax, Logic, Runtime Errors
Debugging Process

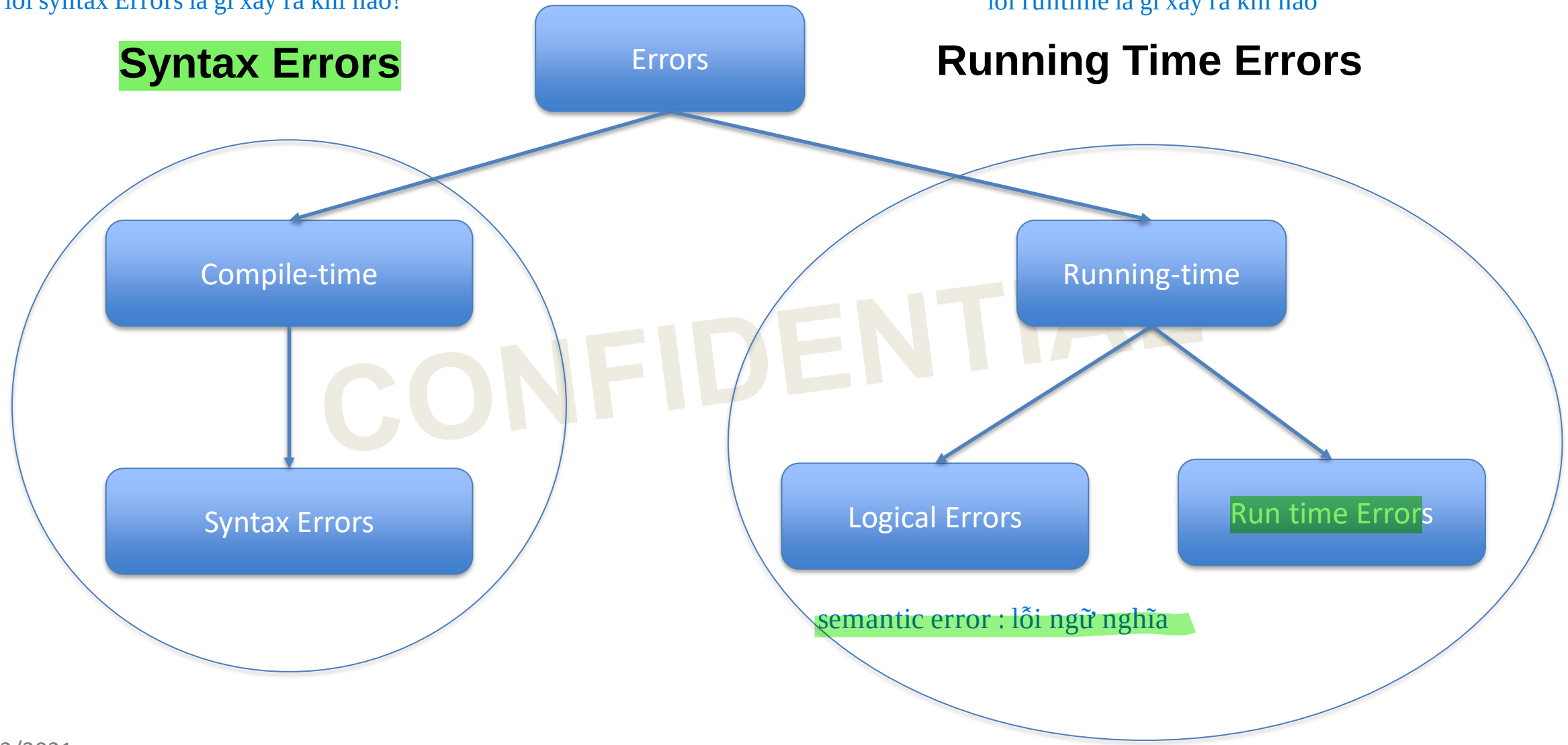
Syntax Errors and Semantic Errors

lỗi syntax Errors là gì xảy ra khi nào?

Syntax Errors

lỗi runtime là gì xảy ra khi nào

Running Time Errors



semantic error : lỗi ngữ nghĩa

Syntax Errors: Define

lỗi cú pháp: định nghĩa

xảy ra

bị sử dụng sai

- Syntax Errors are occurred when rules of the program is misused i.e, when grammatical rule of C++ is violated.

bị vi phạm

Lỗi cú pháp xảy ra khi các quy tắc của chương trình bị sử dụng sai, tức là khi quy tắc ngữ pháp của C++ bị vi phạm.

- Ex: int a, b (semicolon missing)
- Program cannot translate a program into executable code if one of syntax error is present.

Trình biên dịch không thể chuyển đổi chương trình thành mã thực thi nếu có một lỗi cú pháp nào đó.

Syntax Errors: How Detect and Solution

Cách phát hiện và giải pháp

- The compiler detects them when you try to compile your program.

  Trình biên dịch phát hiện chúng khi bạn cố gắng biên dịch chương trình của mình.

- Solution: Find the line(s) containing the syntax errors using the compiler's flagged lines.

  Giải pháp: Tìm dòng (các dòng) chứa lỗi cú pháp bằng cách sử dụng các dòng được trình biên dịch đánh dấu

Syntax Errors: Common Examples

- Missing Semicolon

```
int main(int argc, char *argv[])
{
    int num;
    float value
    double bigNum;
    return 0;
}
```

```
expected initializer before 'double'
expected ';' at end of declaration
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
```

- Undeclared Variable Name – Incorrect Name Of Variable

```
#ifdef SYNTAX2
    int num;
    float value;
    double bigNum;
    bignum = num + value;
#endif
```

```
use of undeclared identifier 'bignum'
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
'bignum' was not declared in this scope
```

Syntax Errors: Common Examples

- Undeclared Variable Name – Missing Semicolon Before That Variable

```
#ifdef SYNTAX2
    int num;
    float value
    double bigNum;
    bigNum = num + value;
#endif
```

'bigNum' was not declared in this scope
'value' was not declared in this scope

- Undeclared Variable Name – Missing Reference to Namespace or include

```
#include <string>
//#define SYNTAX1;
//#define SYNTAX2_1;
//#define SYNTAX2_2;
#define SYNTAX2_3;
int main(int argc, char *argv[])
{
#ifdef SYNTAX2_3
    string s;
#endif
}
```

'string' was not declared in this scope
string s;
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
unknown type name 'string'

Syntax Errors: Common Examples

- Undeclared Variable Name – Using local variable out of scope

```cpp
#include <iostream>

int main(int argc, char *argv[])
{
    {
        int i = 2;
    }
    std::cout << i << std::endl; // i is not in the scope of the main function
    return 0;
}
```

```
error C2065: 'i': undeclared identifier
```

- Not Found a Function– Using but not declared khai báo

```cpp
#include <iostream>

int main(int argc, char *argv[])
{
    doCompile();
    return 0;
}

void doCompile()
{
    std::cout << "No!" << std::endl;
}
```

C3861 'doCompile': identifier not found

```cpp
#include <iostream>

void doCompile(); // forward declare the function

int main(int argc, char *argv[])
{
    doCompile();
    return 0;
}

void doCompile()
{
    std::cout << "No!" << std::endl;
}
```

Syntax Errors: Common Examples

- Unmatched Parentheses

dấu ngoặc

```
#ifdef SYNTAX3
    int result, firstVal, secondVal, factor;
    result = (firstVal - secondVal / factor;
#endif
}
```

expected ')'
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
expected ')' before ';' token

- Unterminated Strings

chưa kết thúc

```
#ifdef SYNTAX4
    std::string s = "My string;
#endif
```

phép gán

- Left-Hand Side of Assignment does not Contains an L-Value

```
#ifdef SYNTAX5
    double x = 2.0, y = 3.1415, product;
    x * y = product;
#endif
```

expression is not assignable
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
lvalue required as left operand of assignment

Syntax Errors: Common Examples

- Passing Parameter to Function not Correct (not correct type or number parameter)

tham số

```
#define SYNTAX6;
void testFunc(std::string value)
{

}
int main(int argc, char *argv[])
{
#ifdef SYNTAX6
    int x = 5;
    testFunc(x);
#endif
}
```

```
could not convert 'x' from 'int' to 'std::__cxx11::string {aka std::__cxx11::basic_string<char>}'
    testFunc(x);
                ^
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
no matching function for call to 'testFunc'
```

Syntax Errors: Common Examples

- Passing Parameter to Function not Correct (not correct type or number parameter)

```
#define SYNTAX6;
void testFunc(std::string value)
{

}
int main(int argc, char *argv[])
{
#ifdef SYNTAX6
    int x = 5;
    testFunc(x);
#endif
}
```

```
could not convert 'x' from 'int' to 'std::__cxx11::string {aka std::__cxx11::basic_string<char>}'
    testFunc(x);
                ^
/home/kiemnv1/WorkSpace/SyntaxErrorExampler/main.cpp
no matching function for call to 'testFunc'
```

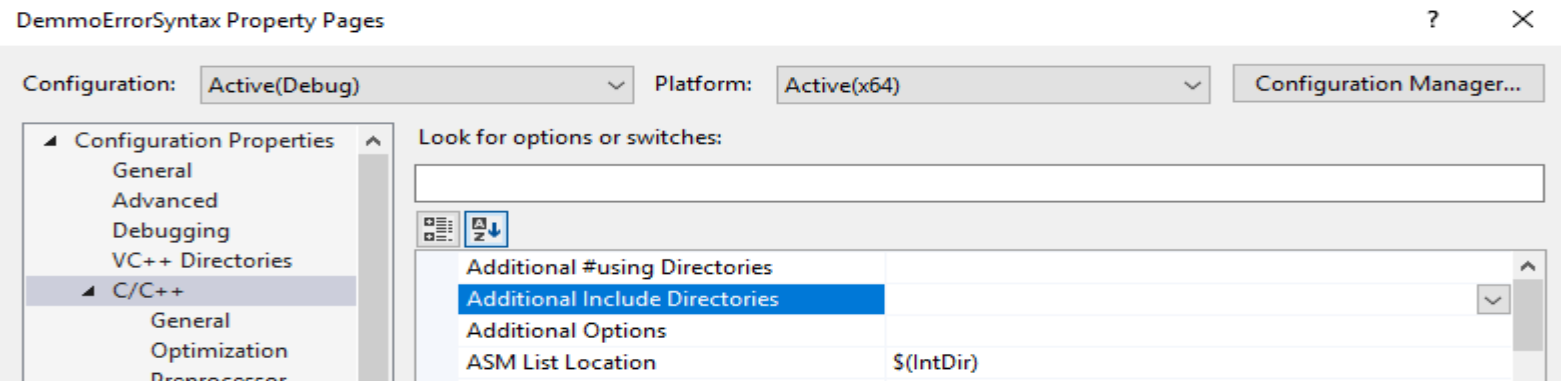
Syntax Errors: Common Examples

- No such file or directory – include file not correct file name or not configure include path

E1696	cannot open source file "myheader.h"	DemmoErrorSyntax	DemmoErrorSyntax.cpp	2
E0020	identifier "doCompile" is undefined	DemmoErrorSyntax	DemmoErrorSyntax.cpp	11
C1083	Cannot open include file: 'myheader.h': No such file or directory	DemmoErrorSyntax	DemmoErrorSyntax.cpp	2

To fix problem we must checking file name included is correct or not. If it is true then checking we have already configure path or not if we use additional header files.

Demo for visual studio:



Runtime Errors: Common Examples

lỗi runtime: các ví dụ phổ biến

- Division by Zero chia cho số 0

```
#ifdef RUNTIME_ERROR_1
    int x = 5;
    int y = 0;
    int z = x/y;
    while (true) {
        std::cout<<"test";
    }
#endif
```

- Accessing at Index Out Of Range of Array, list, vector

truy cập vào chỉ số ngoài phạm vi của mảng, danh sách, vector

```
#ifdef RUNTIME_ERROR_2
    using namespace std;
    std::vector<int> array;
    array.push_back(3);
    array.push_back(4);
    for (int i = 0; i < 8; i++)
    {
        int value = array.at(i);
    }
    while (true) {
        std::cout<<"test";
    }
#endif
```

Runtime Errors: Common Examples

- Accessing a Null Pointer truy cập vào con trỏ null

```
#ifdef RUNTIME_ERROR_3
    int* p = new int;
    delete p;
    p = nullptr;
    using namespace std;
    cout<<"value of p:"<<*p;
    while (true) {
        std::cout<<"test";
    }
#endif
```

- Delete a Pointer but It is not Dynamic Allocate xóa một con trỏ nhưng nó chưa từng được cấp phát động

```
#include <cstdlib>
#include <stdio.h>

void foo()
{
    int a;
    int *p = &a;
    *p = 1;
    printf("*p = %d\n", *p);
    free(p);
}

int main()
{
    foo();
    return 0;
}
```

Logical Errors

- Errors which occur during program execution(run-time) after successful compilation but desired output is not obtained.

đầu ra mong muốn

không đạt được

Lỗi logic là những lỗi xảy ra trong quá trình thực thi chương trình (runtime) sau khi biên dịch thành công nhưng đầu ra mong muốn không được đạt được.

- These errors solely depend on the logical thinking of the programmer and are easy to detect if we follow the line of execution and determine why the program takes that path of execution. We must recheck source code to find the point.

thực thi

xác định

Những lỗi này phụ thuộc hoàn toàn vào tư duy logic của người lập trình và thường dễ phát hiện nếu ta theo dõi dòng thực thi và xác định tại sao chương trình thực hiện theo con đường đó.
Để tìm ra điểm lỗi, ta cần kiểm tra lại mã nguồn và xác định vị trí lỗi.

Logical Errors: Common Examples

- Infinite Loop vòng lặp vô hạn

```
#ifdef LOGIC_1
    using namespace std;
    int i = 0;
    while (i < 10) {
        if (i == 2)
        {
            continue;
        }
        i++;
    }
    cout<<"Exnited loop";
#endif
```

- Misunderstanding of Operator Precedence hiểu sai ưu tiên của toán tử

```
#ifdef LOGIC_2
    int x= 4, y =9, z = 10;
    // value =  (x-y)/z;
    int value = x - y / z;
#endif
```

Logical Errors: Common Examples

- Dangling Else

thiếu dấu ngoặc nhọn sau mệnh đề else để chỉ rõ đoạn mã nào thuộc về mệnh đề if, mệnh đề else

```
#ifdef LOGIC_3
    bool relative = false;
    bool myFriend = true;
    if (relative)
        if (myFriend)
            std::cout <<"both";
    else
        { std::cout << "Not related"; }
    while (true) {

    }
#endif
```

- Missing break on Switch …case

thiếu break sau mỗi case - khi đó sau khi thực hiện 1 case thỏa mãn nó sẽ chạy xuống thực hiện case tiếp theo

```
int x = 2;
switch (x) {
case 1:
{
    qDebug()<<"Mot";
    break;
}
case 2:
{
    qDebug()<<"Hai";
}
case 3:
{
    qDebug()<<"Ba";
}
default:{
    break;
}
}
```

Logical Errors: Common Examples

- Using Assignment Operator on Check Condition

sử dụng toán tử gán trong điều kiện kiểm tra

```
#ifdef LOGIC_5
    int x = 2;
    if (x = 1)
    {
        std::cout<<"x == 1";
    }
#endif
```

- Define a Variable but Not Initial Value

```
int
sum(int *A, int n)
{
  int x, i = 0;

  while(i < n) {
    x = x + A[i];
    ++i;
  }
  return x;
}
```

x is not initialized

định nghĩa biến nhưng không khởi tạo giá trị ban đầu,
=> khi đó giá trị của biến có thể ngẫu nhiên phụ thuộc vào giá trị vùng nhớ tồn tại trước đó

Chú ý:

Logical Errors: How To Prevent

- Checking warning out put when compile program (ensure –Wall flag is on)

sử dụng -Wall để kiểm tra các cảnh báo có thể gây ra lỗi cho chương trình

```
Caos-MacBook-Pro:Desktop vietcv.fho$ g++ -o test test.cpp
Caos-MacBook-Pro:Desktop vietcv.fho$ g++ -Wall -o test test.cpp
test.cpp:8:13: warning: variable 'a' is uninitialized when used here
      [-Wuninitialized]
    cout << a << endl;
            ^
test.cpp:7:10: note: initialize the variable 'a' to silence this warning
    int a;
         ^
          = 0
1 warning generated.
```

- Using –Werror flag to change warnings to errors

sử dụng Werror để chuyển các cảnh báo thành lỗi

```
Caos-MacBook-Pro:Desktop vietcv.fho$ g++ -Wall -Werror -o test test.cpp
test.cpp:8:13: error: variable 'a' is uninitialized when used here
      [-Werror,-Wuninitialized]
    cout << a << endl;
            ^
test.cpp:7:10: note: initialize the variable 'a' to silence this warning
    int a;
         ^
          = 0
1 error generated.
```

Debugging: Basic and Purpose

mục đích

- Debugging is process to correct runtime errors and logical errors.

là quá trình

Debugging là quá trình sửa lỗi trong chương trình máy tính, bao gồm cả sửa lỗi thời gian chạy và lỗi logic.

- In software development, debugging involves locating and correcting code errors in a computer program.

Trong phát triển phần mềm, debugging là quá trình quan trọng nhằm xác định và khắc phục các lỗi trong mã nguồn của chương trình.

Debugging: Strategy chiến lược

We have some ways to debug a program:

1. Review source code and checking logic of code. Xem lại mã nguồn và kiểm tra logic của mã.
2. Run program again and try to reproduce issue. Try to find out the condition that issue happens to reduce scope of source code cause error. After find of source code area can cause effect we can comment out to confirm.

   tái tạo vấn đề điều kiện thu hẹp phạm vi

   Chạy lại chương trình và cố gắng tái tạo vấn đề. Cố gắng tìm ra điều kiện khi vấn đề xảy ra để thu hẹp phạm vi của mã nguồn gây lỗi. Sau khi xác định được khu vực mã nguồn có thể gây ra vấn đề, chúng ta có thể tạm thời bỏ dòng lệnh này để xác nhận.
3. Printing the value of variable. In ra giá trị của biến.
4. Writing the info of variable to log file if we can't check during program running.
5. Using debug tool: gdb, debugger plugin on IDE.

Ghi thông tin của biến vào tệp nhật ký nếu chúng ta không thể kiểm tra trong quá trình chạy chương trình.

Sử dụng công cụ gỡ lỗi như gdb, plugin gỡ lỗi trên IDE.

Debugging: Write Log to File

An alternative approach to conditionalized debugging via the preprocessor is to send your debugging information to a log file. A **log file** is a file (normally stored on disk) that records events that occur in software. The process of writing information to a log file is called **logging**. Most applications and operating systems write log files that can be used to help diagnose issues that occur.

chuẩn đoán xảy ra

We can write functions to write log or using another third party, ex: plog

```
#include <iostream>
#include <plog/Log.h> // Step 1: include the logger header

int getUserInput()
{
    LOGD << "getUserInput() called"; // LOGD is defined by the plog library

    std::cout << "Enter a number: ";
    int x{};
    std::cin >> x;
    return x;
}

int main()
{
    plog::init(plog::debug, "Logfile.txt"); // Step 2: initialize the logger

    LOGD << "main() called"; // Step 3: Output to the log as if you were wri
ting to the console

    int x{ getUserInput() };
    std::cout << "You entered: " << x;

    return 0;
}
```

có thể khởi tạo luôn 1 file txt: và khi viết function thì đồng thời viết giá trị vào file đó

```
2018-12-26 20:03:33.295 DEBUG [4752] [main@14] main() called
2018-12-26 20:03:33.296 DEBUG [4752] [getUserInput@4] getUserInput() called
```

Một phương pháp thay thế cho việc gỡ lỗi có điều kiện thông qua trình biên dịch tiền xử lý là gửi thông tin gỡ lỗi của bạn vào một tệp log. Tệp log là một tệp (thường được lưu trữ trên đĩa cứng) ghi lại các sự kiện xảy ra trong phần mềm. Quá trình ghi thông tin vào tệp log được gọi là logging. Hầu hết các ứng dụng và hệ điều hành đều ghi các tệp log mà có thể được sử dụng để giúp chẩn đoán các vấn đề xảy ra.
Chúng ta có thể viết các hàm để ghi log hoặc sử dụng các công cụ bên thứ ba khác như plog.

Debugging: Debugger trình gỡ lỗi

What can a debugger help you with ?

1. Follow execution flow of an application.
2. Find out which function call which function.
3. Checking current value of variable.
4. Get a stack trace when the application crashed.
5. Investigate why an application hangs.

treo

Trình gỡ lỗi có thể giúp bạn như thế nào?

1.Theo dõi luồng thực thi của một ứng dụng.
2.Xác định hàm nào gọi hàm nào.
3.Kiểm tra giá trị hiện tại của biến.
4.Nhận một ngăn xếp gọi khi ứng dụng gặp sự cố.
5.Khám phá lý do tại sao một ứng dụng bị treo.

Debugging: Debugger

How Can You Use Debugger ?

- Using IDE support debugger tool., Ex for Visual Studio:

https://docs.microsoft.com/en-us/visualstudio/debugger/getting-started-with-the-debugger-cpp?view=vs-2019

- Using gdb by command:

https://mathhoang.blogspot.com/2010/12/huong-dan-debug-chuong-trinh-voi-gdb.html

http://www.gdbtutorial.com/gdb_commands

Thank you