



Bài 2: Advanced Function Concepts and Debugging

-  **Variadic Functions** – Hàm có số lượng tham số biến đổi được
-  **Assert** – Kiểm tra lỗi với assert

2.1 Variadic Function-Hàm có số lượng tham số thay đổi được

- Để làm được hàm có số lượng tham số đổi được, thì cần tạo biến danh sách tham số kiểu `va_list` : đây là 1 struct đã được định nghĩa trong thư viện `stdarg.h`
- Ưu điểm của `va_list` là bạn có thể thêm mới bao nhiêu tham số vào list cũng được, và tham số này có kiểu khác nhau hoàn toàn nhau cũng được.

Dùng Thư viện `stdarg.h`:

- `va_list`: Tạo danh sách tham số mà số lượng có thể thay đổi được.
- `va_start(args, tham số 1)`: Bắt đầu tạo danh sách tham số. với tham số đầu là tham số 1. Khi khởi tạo list bắt buộc phải truyền vào tham số đầu tiên trước. các tham số 2,3,4 tiếp theo sẽ được khởi tạo tùy ý theo tính toán của bạn với kiểu dữ liệu bất kỳ.
- `va_arg (args, kiểu dữ liệu) :` Tạo một tham số mới cho danh sách `args`. Ép kiểu dữ liệu cho tham số đó theo ý bạn. Trong lúc tạo tham số mới, bạn có thể thực hiện ngay các công việc khác liên quan đến tham số như `printf` dữ liệu ra, cho vào hàm điều kiện...
- `va_end`: Kết thúc tạo danh sách tham số. Nó cần được gọi trước khi kết thúc hàm.
- `...` : Đại diện cho danh sách tham số tiếp theo chưa xác định
- `_VA_ARGS_` : Hiển thị ra danh sách các tham số đó.

Ví dụ 1:

```
// Creat Ham hien thi danh sach doi so bat ki luc them vao
#include <stdio.h>
#include <stdarg.h>

void display(int count, ...) { // ham hien danh sach tham so, tham so 1 = so
phần tử của danh sách
    va_list args;              // tạo danh sách tham số args
    va_start(args, count);      // bắt đầu tạo danh sách, tham số đầu = số
lượng đối số.
    printf("value 1 la %d\n", va_arg(args, int));
    printf("value 2 la %d\n", va_arg(args, double));
    printf("value 3 la %d\n", va_arg(args, int));
```

```

    printf("value 4 la %d\n", va_arg(args, char*)); // Có thể ép kiểu dữ liệu
bất kỳ cho từng tham số, tuy nhiên kết quả in ra sai nếu không đúng kiểu của
nó
    printf("value 51 la %d\n", va_arg(args, char*));

    for (int i = 0; i < count; i++) {
        printf("Value at %d: %d\n", i, va_arg(args, int)); // 1st %d = i, 2rd
%d = va_args(args, int)
    }
    va_end(args);
}

int main()
{
    display(5, 5, 8, 15, 10, 13); // tham số 1 =5 : là số lượng tham số thêm
vào, 5...13: lần lượt là các tham số mới.

    // khi ghi vào danh sách đôi số stdarg sẽ cấp phát vùng nhớ cho từng đôi
số. ví dụ
    // 5: 0x00 -> 0x07
    // 8: 0x07 -> 0x0F
    // Khi gọi va_arg : nó sẽ truy cập từng vùng nhớ để lấy ra giá trị đôi số.
    return 0;
}

```

EX2:

```

// Hàm tính tổng danh sách bất kỳ
#include <stdio.h>
#include <stdarg.h>

int sum(int count, ...) {
    va_list args; // Tạo danh sách tham số args
    va_start(args, count); // tham số 1 = count = số lượng phần tham số

    int result = 0;
    for (int i = 0; i < count; i++) {
        result += va_arg(args, int);
    }

    va_end(args);

    return result;
}

int main() {
    printf("Sum: %d\n", sum(4, 1, 2, 3, 6));
}

```

```
    return 0;
}
```

EX3:

```
// Prog tạo ra 1 struct Data, lấy struct đó làm tham số cho va_list và in ra
danh sách struct đó với số lượng tùy chọn

#include <stdio.h>
#include <stdarg.h>

typedef struct Data
{
    int x;
    double y;
} Data;      //Tạo Struct Data 2 thành viên x,y có kiểu khác nhau int,
double.

void display(int count, ...) {

    va_list args; // tạo danh sách args

    va_start(args, count); // Bắt đầu tạo danh sách, tham số 1 = count = số
lượng tham số thêm vào.

    int result = 0;

    for (int i = 0; i < count; i++)
    {
        Data tmp = va_arg(args, Data); // Tạo các tham số có kiểu Data và gán
vào biến tmp để in ra.
        printf("Data.x at %d is: %d\n", i, tmp.x); //1st %d = i, 2nd %d =tmp.x
        printf("Data.y at %d is: %f\n", i, tmp.y);
    }

    va_end(args);
}

int main() {

    display(3, (Data){2,5.0} , (Data){10,57.0}, (Data){29,36.0});
    return 0;
}
```

Kết quả in

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code
[Running] cd "e:\3.ELECTRONIC\HALA-EMBED PROGRAMMING_GOOD\C & C++ ADVANCE\EXERCISE\Bai 2_ STDARG - ASSERT\" && gcc stdarg_lib_struct.c
stdarg_lib_struct && "e:\3.ELECTRONIC\HALA-EMBED PROGRAMMING_GOOD\C & C++ ADVANCE\EXERCISE\Bai 2_ STDARG - ASSERT\"stdarg_lib_struct
Data.x at 0 is: 2
Data.y at 0 is: 5.000000
Data.x at 1 is: 10
Data.y at 1 is: 57.000000
Data.x at 2 is: 29
Data.y at 2 is: 36.000000
```

EX4: Chương trình đọc dữ liệu từ nhiều cảm biến

```
// Prog đọc dữ liệu từ nhiều cảm biến
#include <stdio.h>
#include <stdarg.h>

typedef enum { // kiểu enum = enumeration là 1 tập hợp hằng số có tên gọi, đại
diện cho 1 nhóm giá trị h
    TEMPERATURE_SENSOR, // Nếu chưa gán giá trị thì mặc định TEMPRATURE_SENSOR
= 0, PRESSURE_SENSOR = 1
    PRESSURE_SENSOR
} SensorType;

void processSensorData(SensorType type, ...) { // Hàm đọc dữ liệu nhiều cảm
biến, tham số 1 là kiểu cảm biến, ...: các tham số tiếp theo chưa xác định
    va_list args;
    va_start(args, type); // Bắt đầu tạo danh sách tham số

    switch (type) {
        case TEMPERATURE_SENSOR: {
            int numArgs = va_arg(args, int);
            int sensorId = va_arg(args, int);
            float temperature = va_arg(args, double);
            printf("Temperature Sensor ID: %d, Reading: %.2f degrees\n",
sensorId, temperature); // 1st %d = sensorId, 2rd %.2f = temperature
            if (numArgs > 2) { // Nếu số lượng tham số >2, Xử lý thêm tham số
nếu có
                char* additionalInfo = va_arg(args, char*);
                printf("Additional Info: %s\n", additionalInfo);
            }
            break;
        }
        case PRESSURE_SENSOR: {
            int numArgs = va_arg(args, int);
            int sensorId = va_arg(args, int);
            int pressure = va_arg(args, int);
            printf("Pressure Sensor ID: %d, Reading: %d Pa\n", sensorId,
pressure);
            if (numArgs > 2) {
                // Xử lý thêm tham số nếu có
                char* unit = va_arg(args, char*);
            }
        }
    }
}
```

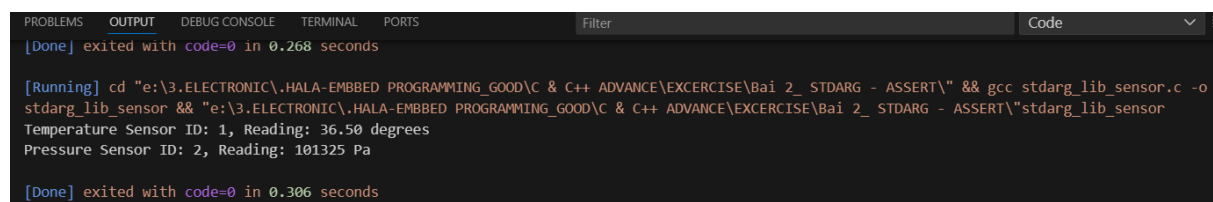
```

        printf("Unit: %s\n", unit);
    }
    break;
}
}

va_end(args); // kết thúc việc tạo tham số cho list.
}

int main() {
    processSensorData(TEMPERATURE_SENSOR, 2, 1, 36.5, "Room Temperature");
    processSensorData(PRESSURE_SENSOR, 2, 2, 101325);
    return 0;
}

```



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter Code
[Done] exited with code=0 in 0.268 seconds

[Running] cd "e:\3.ELECTRONIC\HALA-EMBED PROGRAMMING_GOOD\C & C++ ADVANCE\EXERCISE\Bai 2_ STDARG - ASSERT\" && gcc stdarg_lib_sensor.c -o stdarg_lib_sensor && "e:\3.ELECTRONIC\HALA-EMBED PROGRAMMING_GOOD\C & C++ ADVANCE\EXERCISE\Bai 2_ STDARG - ASSERT\"stdarg_lib_sensor
Temperature Sensor ID: 1, Reading: 36.50 degrees
Pressure Sensor ID: 2, Reading: 101325 Pa

[Done] exited with code=0 in 0.306 seconds

```

EX5: Chương trình tạo lệnh điều khiển nhiều thiết bị với ID khác nhau

```

// Pro tạo các lệnh điều khiển (Turn on/off, set level, send message) cho
nhiều thiết bị theo ID từng thiết bị.
#include <stdio.h>
#include <stdarg.h>

typedef enum { // kiểu enum = enumeration là 1 tập hợp hằng số có tên gọi,
đại diện cho 1 nhóm giá trị. thường dùng vs hàm switch case
    TURN_ON,
    TURN_OFF,
    SET_LEVEL,
    SEND_MESSAGE
} CommandType; // Tao struct lenh dieu khien (bật, tắt, cài mức, gửi thông
diệp)
void sendCommand(CommandType command, ...) {
    va_list args; // tạo list tham số args.
    va_start(args, command); // Bắt đầu tạo list tham số, tham số đầu gán =
command.

    switch (command) {
        case TURN_ON:
        case TURN_OFF: {
            int deviceID = va_arg(args, int); // tạo biến id thiết bị
            printf("Command: %s Device ID: %d\n", command == TURN_ON ? "Turn
On" : "Turn Off", deviceID);

```

```

        break;
    /*
    %s = command == TURN_ON ? "Turn On" : "Turn Off" : là biểu thức có
điều kiện
    %d = deviceID
    command == TURN_ON ? "Turn On" : "Turn Off":
    Đây là một biểu thức điều kiện (ternary operator) kiểm tra giá trị của
biến command.
    Nếu command bằng TURN_ON, biểu thức sẽ trả về chuỗi "Turn On".
    Ngược lại, biểu thức sẽ trả về chuỗi "Turn Off".

    */
}
case SET_LEVEL: {
    int deviceID = va_arg(args, int); // tạo tham số = id máy
    int level = va_arg(args, int); // tham số = level, ép kiểu int
    printf("Set Level of Device ID %d to %d\n", deviceID, level);
    break;
}
case SEND_MESSAGE: {
    char* message = va_arg(args, char*);
    printf("Send Message: %s\n", message); // in ra thông điệp
    break;
}
}

va_end(args);
}

int main() {
    sendCommand(TURN_ON, 1);
    sendCommand(TURN_OFF, 2);
    sendCommand(SET_LEVEL, 3, 75);
    sendCommand(SEND_MESSAGE, "Hello World");
    return 0;
}

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

Filter

```

[Running] cd "e:\3.ELECTRONIC\HALA-EMBBED PROGRAMMING_GOOD\C & C++ ADVANCE\E
stdarg_lib_on_off_device && "e:\3.ELECTRONIC\HALA-EMBBED PROGRAMMING_GOOD\C
Command: Turn On Device ID: 1
Command: Turn Off Device ID: 2
Set Level of Device ID 3 to 75
Send Message: Hello World

```

[Done] exited with code=0 in 0.284 seconds

2.2. Thư viện assert

- Cung cấp macro assert.
- Macro này được sử dụng để kiểm tra một điều kiện.
- Nếu điều kiện đúng (true), không có gì xảy ra và chương trình tiếp tục thực thi.
- Nếu điều kiện sai (false), chương trình dừng lại và thông báo một thông điệp lỗi.
- Dùng trong debug, dùng `#define NDEBUG` để tắt debug

Bản chất assert cũng giống dùng if else nhưng if else không cho biết được lỗi đó nằm ở dòng nào, file nào nhưng assert thì thông báo rõ nó nằm ở file nào, line nào.

Assertion failed: x==5, file assert_lib_intro.c, line 5

```
1 #include <stdio.h>
2 #include <assert.h>
3 int main()
4 {
5     int x = 56;
6     assert(x==5); // macro kiểm tra điều kiện x =5 nếu không thỏa mãn sẽ báo lỗi và nó sẽ ghi rõ ở file nào và dòng nào để bạn dễ tìm.
7     printf("X is: %d",x);
8     return 0;
}
```

[Done] exited with code=3221226505 in 0.389 seconds

[Running] cd "e:\3.ELECTRONIC\...&& gcc assert_lib_intro.c -o assert_lib_intro && "e:\3.ELECTRONIC\...&& assert_lib_intro

Assertion failed: x==5, file assert_lib_intro.c, line 5