Bài 2: STDARG - ASSERT

Phan Hoàng Trung

Thư viện stdarg

- Cung cấp các phương thức để làm việc với các hàm có số lượng input parameter không cố định.
- Các hàm như printf và scanf là ví dụ điển hình

Thư viện stdarg

- va_list: là một kiểu dữ liệu để đại diện cho danh sách các đối số biến đổi.
- va_start: Bắt đầu một danh sách đối số biến đổi. Nó cần được gọi trước khi truy cập các đối số biến đổi đầu tiên.
- va_arg: Truy cập một đối số trong danh sách. Hàm này nhận một đối số của kiểu được xác định bởi tham số thứ hai
- va_end: Kết thúc việc sử dụng danh sách đối số biến đổi. Nó cần được gọi trước khi kết thúc hàm.

Thư viện stdarg

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

void display(int count, ...) {
    va_list args;
    va_start(args, count);


    for (int i = 0; i < count; i++) {
        printf("Value at %d: %d\n", i, va_arg(args,int));
    }

    va_end(args);
}

int main()
{
    display(5, 5, 8, 15, 10, 13);

    return 0;
}
```

Thư viện stdarg

0x01	0x05	0x09	0x0D	0x10
5	8	15	10	13

Thư viện stdarg

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

int sum(int count, ...) {
    va_list args;
    va_start(args, count);

    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, 4));
    return 0;
}
```

Thư viện stdarg

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


typedef struct Data
{
    int x;
    double y;
} Data;

void display(int count, ...) {

    va_list args;

    va_start(args, count);

    int result = 0;

    for (int i = 0; i < count; i++)
    {
        Data tmp = va_arg(args,Data);
        printf("Data.x at %d is: %d\n", i,tmp.x);
```

Thư viện stdarg

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

typedef enum {
    TEMPERATURE_SENSOR,
    PRESSURE_SENSOR
} SensorType;

void processSensorData(SensorType type, ...) {
    va_list args;
    va_start(args, type);

    switch (type) {
        case TEMPERATURE_SENSOR: {
            int numArgs = va_arg(args, int);
            int sensorId = va_arg(args, int);
            float temperature = va_arg(args, double); // float được promote thành double
            printf("Temperature Sensor ID: %d, Reading: %.2f degrees\n", sensorId, temperature);
            if (numArgs > 2) {
                // Xử lý thêm tham số nếu có
                char* additionalInfo = va_arg(args, char*);
                printf("Additional Info: %s\n", additionalInfo);
            }
```

Thư viện stdarg

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

typedef enum {
    TURN_ON,
    TURN_OFF,
    SET_LEVEL,
    SEND_MESSAGE
} CommandType;

void sendCommand(CommandType command, ...) {
    va_list args;
    va_start(args, command);

    switch (command) {
        case TURN_ON:
        case TURN_OFF: {
            int deviceID = va_arg(args, int);
            printf("Command: %s Device ID: %d\n", command == TURN_ON ? "Turn On" : "Turn Off", deviceID);
            break;
        }
        case SET_LEVEL: {
            int deviceID = va_arg(args, int);
```

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

Thư viện assert

```c
#include <stdio.h>
#include <assert.h>

int main() {
    int x = 5;

    assert(x == 5);

    // Chương trình sẽ tiếp tục thực thi nếu
điều kiện là đúng.
    printf("X is: %d", x);

    return 0;
}
```

Thư viện assert

- Lỗi truy cập mảng không an toàn.
- Lỗi chia cho số 0.
- Chia số nguyên cho số nguyên, kết quả là số thực.

Thư viện assert

```
// Macro dùng để debug
#define LOG(condition, cmd) assert(condition && #cmd)
```

Thư viện assert

```
#include <assert.h>

#define ASSERT_IN_RANGE(val, min, max) assert((val) >=
(min) && (val) <= (max))

void setLevel(int level) {
    ASSERT_IN_RANGE(level, 1, 10);
    // Thiết lập cấp độ
}
```

Thư viện assert

```
#include <assert.h>
#include <stdint.h>

#define ASSERT_SIZE(type, size) assert(sizeof(type) ==
(size))

void checkTypeSizes() {
    ASSERT_SIZE(uint32_t, 4);
    ASSERT_SIZE(uint16_t, 2);
    // Kiểm tra các kích thước kiểu dữ liệu khác
}
```