C++ 강좌 01회 : C 언어에서 중급자들이 많이 사용하는 코드 패턴

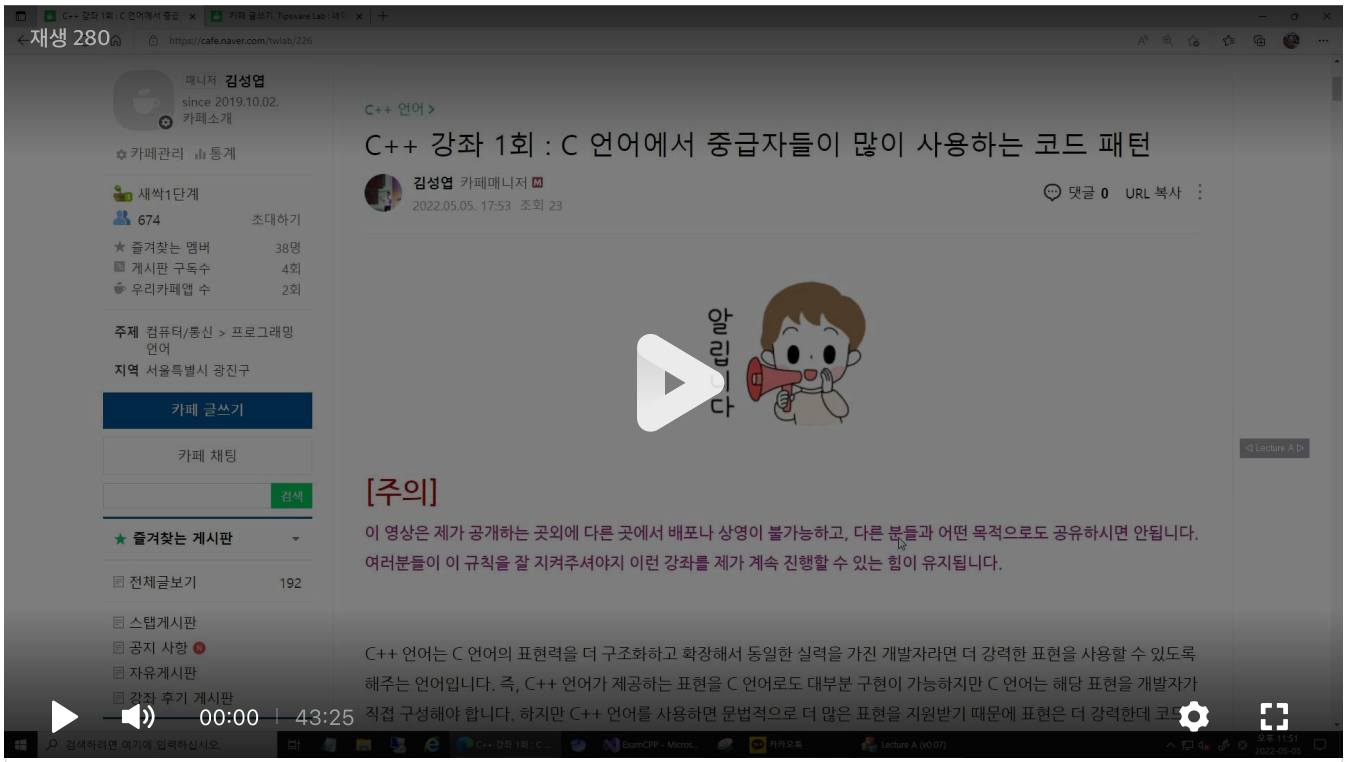
김성엽 카페매니저 + 구독 1:1 채팅
2022.05.05. 17:53 조회 766

[주의]

이 영상은 제가 공개하는 곳외에 다른 곳에서 배포나 상영이 불가능하고, 다른 분들과 어떤 목적으로도 공유하시면 안됩니다. 여러분들이 이 규칙을 잘 지켜주셔야지 이런 강좌를 제가 계속 진행할 수 있는 힘이 유지됩니다.

C++ 언어는 C 언어의 표현력을 더 구조화하고 확장해서 동일한 실력을 가진 개발자라면 더 강력한 표현을 사용할 수 있도록 해주는 언어입니다. 즉, C++ 언어가 제공하는 표현을 C 언어로도 대부분 구현이 가능하지만 C 언어는 해당 표현을 개발자가 직접 구성해야 합니다. 하지만 C++ 언어를 사용하면 문법적으로 더 많은 표현을 지원받기 때문에 표현은 더 강력한데 코드 구성은 더 간결해지는 특징이 있습니다.

그런데 C++ 언어가 제공하는 표현들이 결국 C 언어가 사용하던 표현을 문법적으로 재구성한 것이기 때문에 해당 기능들이 C 언어의 어떤 표현을 응용한 것인지 이해하면 C++ 언어를 좀더 쉽게 이해하고 공감 할 수 있습니다. 그래서 이번 시간에는 C++ 언어 문법을 배우기 전에 C 언어를 사용하다가 왜 C++ 언어를 사용하게 되었는지를 먼저 소개하도록 하겠습니다.



1회 : C 언어에서 중급자들이 많이 사용하는 코드 패턴

1. 강의에 사용할 기본 예제 코드

아래의 코드는 사용자에게 두 개의 수를 입력받아 합산한 결과를 출력하는 예제입니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

int main()
{
    int a, b;

    printf("덧셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 합산된 결과를 출력한다.
    printf("%d + %d = %d\n", a, b, a + b);

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
덧셈할 수를 입력하세요: 3 5
3 + 5 = 8

이 창을 닫으려면 아무 키나 누르세요...
```

2. 코드의 중복을 어떻게 해결할 것인가?

위 예제 코드에서 사용한 정숫값을 입력받고 합산하여 출력하는 기능이 여러 번 반복되어 사용되면 아래와 같이 중복된 코드가 나열될 것입니다. 그런데 이렇게 코드가 나열되면 코드가 길어지는 문제만 있는 것이 아니라 코드에 변화가 생겼을 때 동일한 작업을 여러번 반복해야 하는 불편함도 생깁니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

int main()
{
    int a, b;

    printf("덧셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 합산된 결과를 출력한다.
    printf("%d + %d = %d\n", a, b, a + b);

    printf("덧셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 합산된 결과를 출력한다.
    printf("%d + %d = %d\n", a, b, a + b);

    printf("덧셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 합산된 결과를 출력한다.
    printf("%d + %d = %d\n", a, b, a + b);

    return 0;
}
```

예를 들어, 위 코드에서 덧셈의 기능이 곱셈으로 바뀌면 어떻게 될까요? printf 함수와 주석에 사용한 '덧셈'이라는 문구를 여섯 번 찾아서 '곱셈'으로 변경하고, 출력과 연산에 사용한 덧셈(+) 기호를 곱셈(*) 기호로 여섯 번 변경해야합니다. 즉, 간단한 기능 변경에도 아래와 같이 총 12곳을 수정해야합니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

int main()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");    // 덧셈 -> 곱셈 수정
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.  // 덧셈 -> 곱셈 수정
    printf("%d * %d = %d\n", a, b, a * b);   // + 기호를 * 기호로 2회 수정

    printf("곱셈할 수를 입력하세요: ");    // 덧셈 -> 곱셈 수정
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.  // 덧셈 -> 곱셈 수정
    printf("%d * %d = %d\n", a, b, a * b);   // + 기호를 * 기호로 2회 수정

    printf("곱셈할 수를 입력하세요: ");    // 덧셈 -> 곱셈 수정
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.  // 덧셈 -> 곱셈 수정
    printf("%d * %d = %d\n", a, b, a * b);   // + 기호를 * 기호로 2회 수정

    return 0;
}
```

그래서 프로그래밍 언어는 이런 중복 코드를 줄이는 여러가지 방법을 제공합니다. 먼저 위와 같이 **근접한 위치에서 동일(유사)한 코드가 반복되면 아래와 같이 반복문을 사용**하여 중복코드를 줄입니다. 아래와 같이 반복문을 사용하면 위 예제와 동일하게 세 번 값을 입력받아 곱셈한 결과를 출력하지만 중복코드가 존재하지 않아서 기능적인 변화가 생겼을 때 대처가 단순해 집니다. 즉, **반복 작업 표현은 중복 코드를 줄이는 의미가 포함**되어 있습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

int main()
{
    int a, b;

    for (int i = 0; i < 3; i++) {
        printf("곱셈할 수를 입력하세요: ");
        // 두 개의 정숫값을 입력받는다.
        scanf_s("%d %d", &a, &b);
        // 입력받은 정숫값과 곱셈의 결과를 출력한다.
        printf("%d * %d = %d\n", a, b, a * b);
    }

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
곱셈할 수를 입력하세요: 2 6
2 * 6 = 12
곱셈할 수를 입력하세요: 3 4
3 * 4 = 12
곱셈할 수를 입력하세요: 7 5
7 * 5 = 35

이 창을 닫으려면 아무 키나 누르세요...
```

하지만 모든 중복 코드를 반복문으로 줄일 수 있는 것은 아닙니다. 예를 들어, 아래와 같이 중복된 코드가 여러 곳에 흩어져있는 경우에는 반복문으로 중복 코드를 줄일 수 없습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

int main()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);

    /*
        규칙성 없는 다른 코드가 있다고 가정합니다.
    */

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);

    return 0;
}
```

그래서 이런 경우에는 반복문 대신 함수를 사용해서 아래와 같이 중복 코드를 줄입니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void ShowResult()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    ShowResult();

    /*
        규칙성 없는 다른 코드가 있다고 가정합니다.
    */

    ShowResult();

    return 0;
}
```

물론 함수와 반복문은 서로 배타적인 기술이 아니기 때문에 반복문으로만 구성했던 코드를 아래와 같이 함수와 반복문을 함께 사용해서 구성하기도 합니다. 즉, 명령 구성과 관련된 표현은 개발자의 선택에 따라 달라질 뿐입니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void ShowResult()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    for (int i = 0; i < 3; i++) ShowResult();

    return 0;
}
```

3. 함수는 어떻게 만든 것이 좋은가?

함수는 단순히 중복 코드를 줄이기 위해 사용하는 문법이 아닙니다. **사용자 정의 동사이며 기능 변화에 대처하기 위해 구조화된 언어가 사용하는 기술**입니다. 지금 위 이야기를 이해하지 못한다면 아래에 링크한 자료와 동영상 강좌를 먼저 공부하기 바랍니다.

함수와 연산자는 동사이다.
금배씨 목차: https://blog.naver.com/tipsware/222617024788 이 강좌에서는 C언어 명령문에서 동사의 …
blog.naver.com

함수에 대한 못다 한 이야기 (1)
: C 언어 관련 전체 목차 http://blog.naver.com/tipsware/221010831969 이 글은 함수의 문법적 의미에 …
blog.naver.com

함수에 대한 못다 한 이야기 (2)
: C 언어 관련 전체 목차 http://blog.naver.com/tipsware/221010831969 이 글은 아래에 링크한 글에서 …
blog.naver.com

그래서 위에서 소개했던 ShowResult 함수는 좋은 형식의 함수는 아닙니다. 단순히 일정 코드 패턴을 그룹으로 묶어서 분리시켜놓은 형태일 뿐입니다. 즉, 아래의 코드에서 ShowResult 함수는 함수의 형식만 가졌을뿐 함수로써의 의미는 별로 없습니다. 예를 들어, ShowResult 함수는 사용자에게 값을 입력받아 계산하기 때문에 미리 지정된 상수나 다른 변수에 저장된 값으로는 작업을 할 수 없습니다. 어떻게 보면 ShowResult 함수는 역할이 다른 작업들이 하나로 그룹지어져 작업에 대한 변화나 확장을 할 수 없는 구조로 되어 있습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void ShowResult()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);
    // 입력받은 정숫값과 곱셈의 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    ShowResult();

    return 0;
}
```

따라서 아래와 같이 함수의 기능을 줄여서 단순하게 구성하는 것이 활용도가 높습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void MultiplyValue(int a, int b)
{
    // 전달된 두 수의 곱셈 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);

    MultiplyValue(a, b); // 입력받은 변숫값 사용 가능
    MultiplyValue(2, -6); // 상숫값 사용 가능

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
곱셈할 수를 입력하세요: 3 7
3 * 7 = 21
2 * -6 = -12

이 창을 닫으려면 아무 키나 누르세요...
```

그리고 위와 같이 코드를 구성하면 기능 변화에도 대처하기 좋습니다. 예를 들어, 곱셈을 하는데 피연산자가 음수라면 값을 양수로 변경해서 곱셈을 해야 한다면 아래와 같이 MultiplyValue 함수만 수정해서 대응이 가능합니다. 즉, main 함수에 사용된 코드에는 변화가 생기지 않습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void MultiplyValue(int a, int b)
{
    // a 또는 b 변수가 음수라면 양수로 변경한다.
    if (a < 0) a = -a;
    if (b < 0) b = -b;

    // 전달된 두 수의 곱셈 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);

    MultiplyValue(a, b); // 입력받은 변숫값 사용 가능
    MultiplyValue(2, -6); // 상숫값 사용 가능

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
곱셈할 수를 입력하세요: -3 5
3 * 5 = 15
2 * 6 = 12

이 창을 닫으려면 아무 키나 누르세요...
```

만약, 값의 변경 사항을 확인하고 싶다면 아래와 같이 코드를 구성하면 됩니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void MultiplyValue(int a, int b)
{
    // a 또는 b 변수가 음수라면 양수로 변경한다.
    if (a < 0) printf("a변수의 값을 %d에서 %d로 변경함\n", -a, a = -a);
    if (b < 0) printf("b변수의 값을 %d에서 %d로 변경함\n", -b, b = -b);

    // 전달된 두 수의 곱셈 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);

    MultiplyValue(a, b); // 입력받은 변숫값 사용 가능
    MultiplyValue(2, -6); // 상숫값 사용 가능

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
곱셈할 수를 입력하세요: -3 5
a변수의 값을 -3에서 3로 변경함
3 * 5 = 15
b변수의 값을 -6에서 6로 변경함
2 * 6 = 12

이 창을 닫으려면 아무 키나 누르세요...
```

그리고 위 코드에서 각 변수가 음수이면 양수로 변경하는 코드도 어떻게 보면 중복 코드이기 때문에 아래와 같이 함수를 추가해서 변화에 대처하는 것이 좋습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void ModifyValue(int *p_value, const char *p_var_name)
{
    if (*p_value < 0)
        printf("%s변수의 값을 %d에서 %d로 변경함\n", p_var_name, -*p_value, *p_value = -*p_value);
}

void MultiplyValue(int a, int b)
{
    // a 또는 b 변수가 음수라면 양수로 변경한다.
    ModifyValue(&a, "a");
    ModifyValue(&b, "b");

    // 전달된 두 수의 곱셈 결과를 출력한다.
    printf("%d * %d = %d\n", a, b, a * b);
}

int main()
{
    int a, b;

    printf("곱셈할 수를 입력하세요: ");
    // 두 개의 정숫값을 입력받는다.
    scanf_s("%d %d", &a, &b);

    MultiplyValue(a, b); // 입력받은 변숫값 사용 가능
    MultiplyValue(2, -6); // 상숫값 사용 가능

    return 0;
}
```

4. 함수로 대처가 안되는 기능 변화

함수는 매개 변수가 나열되는 구조로 문법이 구성되기 때문에 매개 변수의 개수가 달라지면 형태적으로 변화가 생겨서 함수를 호출하는 코드까지 모두 변화가 생기는 단점이 있습니다. 예를 들어, 위 예제는 모두 두 개의 정숫값을 사용하지만 기능에 변화가 생겨서 세 개의 정숫값을 사용하도록 변경되면 아래와 같이 MultiplyValue 함수 내부 뿐만아니라 main 함수에서 MultiplyValue 함수를 호출하는 곳까지 모두 변화가 생깁니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

void ModifyValue(int *p_value, const char *p_var_name)
{
    if (*p_value < 0)
        printf("%s변수의 값을 %d에서 %d로 변경함\n",  p_var_name, -*p_value, *p_value = -*p_value);
}

void MultiplyValue(int a, int b, int c)
{
    // a, b 또는 c 변수가 음수라면 양수로 변경한다.
    ModifyValue(&a, "a");
    ModifyValue(&b, "b");
    ModifyValue(&c, "c");

    // 전달된 세 수의 곱셈 결과를 출력한다.
    printf("%d * %d * %d = %d\n", a, b, c, a * b * c);
}

int main()
{
    int a, b, c;

    printf("곱셈할 수를 입력하세요: ");
    // 세 개의 정숫값을 입력받는다.
    scanf_s("%d %d %d", &a, &b, &c);

    MultiplyValue(a, b, c); // 입력받은 변숫값 사용 가능
    MultiplyValue(2, -6, -3); // 상숫값 사용 가능

    return 0;
}
```

위 예제는 아래와 같이 출력됩니다.

```
곱셈할 수를 입력하세요: 3 -7 2
b변수의 값을 -7에서 7로 변경함
3 * 7 * 2 = 42
b변수의 값을 -6에서 6로 변경함
c변수의 값을 -3에서 3로 변경함
2 * 6 * 3 = 36

이 창을 닫으려면 아무 키나 누르세요...
```

그리고 이런 문제는 **매개 변수의 개수가 아닌 매개 변수의 자료형이 바뀌어도 동일한 문제가 발생**합니다. 그래서 C 언어를 사용하는 개발자들은 함수의 매개 변수 구조를 일정한 형식으로 유지하려고 아래와 같이 구조체를 함께 사용합니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

// 함수가 사용할 매개 변수를 새로운 자료형으로 선언
// 따라서 이제 함수의 매개 변수에 변화가 생기면 여기에
// 변화가 생기고 함수의 매개 변수 형식에는 변화가 생기지 않는다.
struct ValueList
{
    int a, b, c;
};

void ModifyValue(int *p_value, const char *p_var_name)
{
    if (*p_value < 0)
        printf("%s변수의 값을 %d에서 %d로 변경함\n", p_var_name, -*p_value, *p_value = -*p_value);
}

void MultiplyValue(struct ValueList values)
{
    // a, b 또는 c 변수가 음수라면 양수로 변경한다.
    ModifyValue(&values.a, "a");
    ModifyValue(&values.b, "b");
    ModifyValue(&values.c, "c");

    // 전달된 세 수의 곱셈 결과를 출력한다.
    printf("%d * %d * %d = %d\n", values.a, values.b, values.c, values.a * values.b * values.c);
}

int main()
{
    struct ValueList values;
    printf("곱셈할 수를 입력하세요: ");
    // 세 개의 정숫값을 입력받는다.
    scanf_s("%d %d %d", &values.a, &values.b, &values.c);
    MultiplyValue(values); // 입력받은 변숫값 사용 가능

    struct ValueList temp = { 2, -6, -3 };  // 상숫값 사용 가능
    MultiplyValue(temp);

    return 0;
}
```

그리고 위 코드에서는 MultiplyValue 함수로 값을 넘길 때 값을 복사하는 형식을 사용하기 때문에 **함수를 호출할 때마다 12 바이트의 메모리 복사가 발생**합니다. 그리고 MultiplyValue 함수에서 값을 복사해서 사용하는 방식을 사용했기 때문에 MultiplyValue 함수 내부에서 음수를 양수로 바꾼 값을 main 함수의 values 변수나 temp 변수에는 영향을 미치지 않습니다. 이것은 **모듈의 독립성은 보장하는 관점에서 본때는 장점**일 수도 있지만 **main 함수에서 MultiplyValue 함수 내부에서 변경된 값은 사용할 수 없다는 단점**이기도 합니다.

이론적인 관점에서는 독립성이 더 중요하지만 실제 프로그래밍에서는 기능 변화나 확장이 중요하고 기능이 효율적으로 실행되는 것이 더 우선시 됩니다. 그래서 개발자들은 값을 복사하는 형태보다는 아래와 같이 포인터 문법을 사용해서 값을 참조

하는 형태로 코드를 많이 구성합니다. 이렇게 구성하면 MyltiplyValue 함수로 매개 변수를 전달할 때, 값이 아닌 주소를 사용하기 때문에 4바이트 크기만 메모리에 복사되고 MyltiplyValue 함수 내에서 수정된 값이 main 함수의 values 변수나 temp 변수에 바로 반영되기 때문에 main 함수에서도 변경된 값을 활용할 수 있습니다.

```c
#include <stdio.h>  // printf, scanf 함수를 사용하기 위해!

// 함수가 사용할 매개 변수를 새로운 자료형으로 선언
// 따라서 이제 함수의 매개 변수에 변화가 생기면 여기에
// 변화가 생기고 함수의 매개 변수 형식에는 변화가 생기지 않는다.
struct ValueList
{
    int a, b, c;
};

void ModifyValue(int *p_value, const char *p_var_name)
{
    if (*p_value < 0)
        printf("%s변수의 값을 %d에서 %d로 변경함\n", p_var_name, -*p_value, *p_value = -*p_value);
}

void MultiplyValue(struct ValueList *p_values)
{
    // a, b 또는 c 변수가 음수라면 양수로 변경한다.
    ModifyValue(&p_values->a, "a");
    ModifyValue(&p_values->b, "b");
    ModifyValue(&p_values->c, "c");

    // 전달된 세 수의 곱셈 결과를 출력한다.
    printf("%d * %d * %d = %d\n", p_values->a, p_values->b, p_values->c,
                    p_values->a * p_values->b * p_values->c);
}

int main()
{
    struct ValueList values;
    printf("곱셈할 수를 입력하세요: ");
    // 세 개의 정숫값을 입력받는다.
    scanf_s("%d %d %d", &values.a, &values.b, &values.c);
    MultiplyValue(&values); // 입력받은 변숫값 사용 가능

    struct ValueList temp = { 2, -6, -3 };  // 상숫값 사용 가능
    MultiplyValue(&temp);

    return 0;
}
```

결국 C 언어로 프로그래밍을 하는 개발자의 코드 패턴 중에 가장 일반적이면서도 좋은 코드 패턴으로 평가받는 형식이 바로 위와 같이 구조체와 함수 그리고 포인터를 사용하는 형태입니다. 하지만 위 코드는 초보자들이 사용하기에는 너무 어렵다는

단점이 있습니다. 즉, C 언어에서 어렵다고 이야기하는 세 가지 문법을 모두 사용하는 형태이기 때문에 좋은 코드 패턴이지만 초보자들에게 가르쳐서 사용하게 만들기는 어렵다는 뜻입니다.

그런데 C++ 언어는 위 표현을 최대한 단순하게 변경해서 기능은 동일하게 유지하지만 초보자들도 이해하고 사용할 수 있는 문법으로 제공합니다. 이것이 C 언어와 C++ 언어의 문법적인 표현 차이가 되기 때문에 다음 강좌에서 이 표현의 차이에 대해 설명하면서 C++ 언어 공부를 시작하도록 하겠습니다.

댓글

GYCCHANG 3

c에서는 cpp에서의 함수 오버로딩이 없어 구조체 매개변수로 구현한다고 했던 cpp온라인모임1기 3회의 내용을 더 잘 이해할 수 있는 너무 소중한 강좌였습니다. c에서의 함수 매개변수에 구조체를 활용하여 매개변수의 개수, 자료형이 달라지는 경우를 대처할 수 있다는 것을 통해 이해하게 되었습니다. 감사드립니다~ㅎㅎ

2022.05.06. 09:50 답글쓰기

김성엽 M 작성자

2022.05.06. 10:34 답글쓰기

정용주

다시 강의를 들으니 오프라인 강의를 들었던 기억이 떠오르네요. 함수는 사용자가 정의한 동사라는 설명을 통해 어떻게 함수를 구현하는 게 좋은지 잘 이해할 수 있었습니다. 또한 c언어에서 함수의 매개변수에 구조체 주소를 넘겨주는 방식을 사용하므로 어떠한 장점이 있는지 생각하게 되었습니다. 감사합니다 ㅎㅎ

2022.05.06. 16:26 답글쓰기

김성엽 M 작성자

파이팅입니다~ :)

2022.05.08. 18:13 답글쓰기

슈퍼울프 V

감사합니다!ㅎㅎ 제목부터 꼭 봐야겠다고 생각했는데..
강의 들으면서 C언어에서 어떻게 추가되는 변수들에 대한 대응 방법이나 데이터의 효율적으로 활용해서 대처하는지 배울 수 있었던 것 같습니다.
쌤 말씀따라.. 10번 반복해야겠네요 ㅎㅎ

2022.05.09. 11:47 답글쓰기

김성엽 M 작성자

ㅎㅎ 그렇지요! 10번 반복 고!

2022.05.09. 11:54 답글쓰기

슈퍼울프 V

안녕하세요 성엽 선생님.

10번 쯤더 반복하다가.. 포인터 응용해서 한번 다른방법으로 덧셈 연산하는 걸 해봤는데요..
아래와 같이 코드가 나오는데..

내용중에.. 포인터룰 while문에 넣어서 Limit을 정해주고 포인터 연산을 진행하도록 바꿔봤는데요..
struct values* p_limit = p_value + 1; 이 표현처럼.. p_value의 메모리 사이즈가.. int a, b, c 로인해서.. 총 12바이트를 가지는데..
p_limt은 거기에 +1을 하게되면, p_limit의 메모리 주소는 +12 메모리 주소를 가지게 되고,
temp에서 4바이트 크기의 메모리 주소를 옮겨가면서 while문을 돌게하면, 총 3번 반복하게 되어서 최종적으로 temp가 가르키는 주소의 값을 총 3번 음수인지 여부를 확인해서 음수이면 변경하도록 했습니다.
쪼금.. 아쉬운 점은.. p_limit을 계산하는 방법을.. p_value +1로 숫자 +1의 상수값이 아닌 다른 방법이 있을 것 같은데.. ㅎㅎ
음... 잘 모르겠습니다.

```
#include < stdio.h>

struct values
{
int a, b, c;
};

void change(struct values* p_value)
{

struct values* p_limit = p_value + 1;

int *temp = p_value;

while (temp < p_limit)
{
if (*temp < 0)
{
*temp = -*temp;
}
temp++;
}

}


void add(struct values *p_value)
{
change(p_value);
printf("a = %d, b = %d, c = %d\n", p_value->a, p_value->b, p_value->c);
printf("%d + %d + %d = %d\n", p_value->a, p_value->b, p_value->c, p_value->a + p_value->b + p_value->c);
}

int main()
{
struct values value;

printf("값을 입력 하세요 : ");
scanf_s("%d %d %d", &value.a, &value.b, &value.c);

add(&value);


return 0;
}
```

2022.05.16. 12:25 답글쓰기

김민기

1. 제 생각은 구조체를 효율적으로 사용하려면 자료형이 다른 데이터를 담아야 한다고 생각합니다.
울프님이 만드신 int 자료형 구조체를 보면 배열로 사용하는 것이 좀 더 효율적이라고 생각합니다.
그리고 이 코드는 int에 다른 자료형이 들어가면 change 함수를 사용할 수 없습니다. 왜냐하면
int를 기반으로 만든 코드이기 때문입니다.
=>서로 다른 자료형을 담기 위해 void형도 생각해 봤지만, void형도 마찬가지로 배열로 선언할 수 있기 때문에 의미가 없다고 생각합니다.

2. p_value+1로 하는것은 딱히 더 좋은 방법이 없습니다! ->sizeof함수를 사용하는 것보다 더 낫기 때문입니다.

3. int*temp=p_value; 코드에서 p_value는 구조체이기 때문에 (int)형변환을 해주어야합니다.

2022.05.16. 13:20 답글쓰기

김태화

김민기 엇 답변 달아 주신 것 이제 확인 했네요.. :)
민기님께서 알려주신 내용을 여러 번 읽어보니.. 또 하나 배웠네요.. :) 감사합니다. ㅎㅎ

1. int 자료형에 해당하는 것만 데이터로 받으려면 배열이 맞겠네요.. ^^ int a, b만 생각하고 연습한다고 생각하고 해본거라서.. 다양한 변수에 대해서는 사용 할 수 없게 코딩 해 버렸네요.. 좀더 고려해서 수정해 봐야 할 것 같네요. ㅎ

2. 저두 sizeof를 이용해서 하는 방법이 있기는 할 것 같은데.. 그것보다는 주소연산으로 해 보는게 좋을것 같다고 생각이 들었습니다. ㅎㅎ

3. p_value는 구조체이기는 하지만,, 그 구조체의 주소를 담고 있고, 32비트라서 자료형이 int라고 생각하고 딱히 정의하지 않고 넘어간건데.. 보다 확실히 하려면 int로 형변환해서.. 컴파일러가 정확하게 인지 하도록 해야겠네요.. ㅎㅎ

답변 달아주셔서 고맙습니다. ㅎ

2022.05.19. 15:00 답글쓰기

fwangs

계속 같은 강의를 반복해서 듣는것도 너무 도움되는데 지속적으로 업데이트 된 강의를 올려주셔서 그저 감사할 따름입니다. 마음을 다잡고 부지런히 따라가 보겠습니다^^ 감사합니다!!!!

2022.05.30. 08:47 답글쓰기

김성엽 작성자

먼서 시작한 선배 개발자가 할일이라고 생각합니다 ㅎ

2022.05.30. 08:49 답글쓰기

fwangs

김성엽 대표님께서 보여주시는 마음가짐까지도 많이 배우겠습니다.:-) 오늘도 건강도 잘 챙기시면서 화이팅입니다!!!

2022.05.30. 08:55 답글쓰기

김성엽 작성자

fwangs 행복한 한주 보내요~ :)

2022.05.30. 09:30 답글쓰기

bac

영상 22:30쯤에 함수의 매개 변수들은 뒤에서부터 앞쪽으로 처리된다고 말씀하셨는데 c/c++의 모든 함수들에 해당하는 사항인가요?

2022.06.23. 15:54 답글쓰기

김성엽 작성자

네~

2022.06.23. 16:03 답글쓰기

bac

김성엽 감사합니다

2022.06.23. 16:10 답글쓰기

Zermi 3

와-
printf에서 "어? 왜 -a로 찍으셨지? 그냥 a 아닌가?" 하고 물어봐야겠다 생각했는데, 바로 다음에 뒤부터 먼저 처리 된다는 거 바로 설명해 주시고,
함수에 대한 정확한 의미?에 대해 다시 생각하게 만들어 주시고,
printf문 2개를 굳이 함수화 해야되나? 였는데 제가 부족하다고 느끼게 해주는 명강의였습니다!

집에 C++책이 2권이나 있어서 그냥 책만 보고 공부할까 했는데 역시 선생님 강의 듣길 잘했다고 생각이 듭니다.
책만 봤으면 아 C++에서는 이렇게 쓰는구나.. 하고 끝났을 내용을 C에서 이랬던 것을 대응하기 위해 C++에서는 이렇게 쓸 것이다! 이렇게 강의 해주시니 다음 강의가 너무 기대됩니다!!
완강하겠습니다!

2022.09.03. 13:33 답글쓰기

김성엽 M 작성자

ㅎㅎ 잘 공부하셔서 주변에 많이 홍보해주세요!

2022.09.03. 13:56 답글쓰기

티모 3

주소를 넘긴다! 반복해봐야겠어요

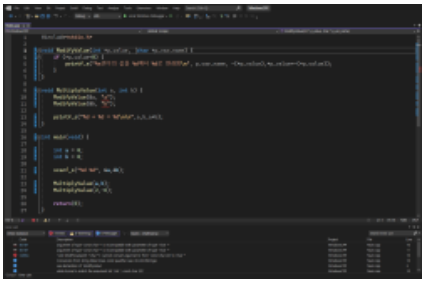
2022.11.07. 20:15 답글쓰기

김성엽 M 작성자

2022.11.07. 21:22 답글쓰기

황금잉어가물치 3

스샷과 같이 ModifyValue함수의 매개변수 중 char *p_var_name가 문자열을 받을 때 const상수 설정을 안 하면 에러가 나는데 이유가 왜 그러는지 알수 있을까요? c++뿐만 아니라 c언어에서도 문자열을 변수로 받으면 무조건 const 설정을 해야하는지도 궁금합니다. 알려주시기 바랍니다.



2022.11.12. 16:34 답글쓰기

김성엽 M 작성자

아래에 링크한 글을 읽어보세요.
https://blog.naver.com/tipsware/221242721220

2022.11.12. 16:39 답글쓰기

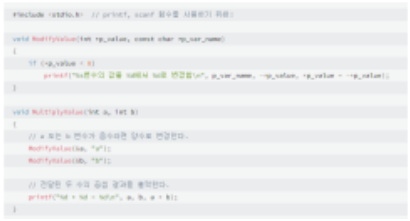
황금잉어가물치 3

김성엽 감사합니다

2022.11.12. 16:53 답글쓰기

황금잉어가물치 3

음수의 값이 넘어오면 if문을 거쳐 음수가 나오는게 아니고 양수가 나오나요? 갑자기 이상해서 질문드립니다. *p_value가 if문을 통해 음수로 되면 참이되면 양수가 되어 printf문에서 2번째 변수 -*p_value로 표현하고 3번째 변수에서 다시한번 -*p_value를 한번 더 써 양수로 바꾸는 건가요? 갑자기 햇갈리네요. 답변 부탁드립니다.



2022.11.13. 15:50 답글쓰기

김성엽 M 작성자

함수를 호출할때 매개 변수는 뒤쪽에서 앞쪽으로 오면서 처리가 됩니다. 따라서 *p_value = -*p_value 부터 처리가 됩니다. 따라서 음수였던 값이 양수로 바뀝니다. 그런데 -5가 5로 변경되었다고 출력해야 하기 때문에 그 앞에 출력되는 값은 양수 값이 -를 붙여서 음수로 출력하는 것입니다.

이 코드가 헷갈리시면 아래와 같이 코드를 나누어서 사용하시면 됩니다.
if(*p_value < 0){
*p_value = -*p_value;
printf("%s변수의 값을 %d에서 %d로 변경함\n", p_var_name, -*p_value, *p_value);
}

2022.11.13. 17:50 답글쓰기

황금잉어가물치 3

김성엽 감사합니다.

2022.11.13. 18:47 답글쓰기

카일 3

강의 잘 들었습니다. 여러번 들어도 들을 때마다 새로운 걸 배우게 됩니다. 감사합니다 ~~

2022.11.15. 03:33 답글쓰기

김성엽 M 작성자

2022.11.15. 10:01 답글쓰기

해피파파 3

ModifyValue(&this->a, "a"); 에서 "a" -> 'a' 로 바꾸고 %s를 %c로 바꿔도 된다라고 설명이 나오는데,
차이점이 궁금하시다면 아래 설명을 꼭 읽어보세요~ 추천합니다~

1) https://blog.naver.com/tipsware/221018307213

2) https://blog.naver.com/tipsware/221242721220

2022.11.21. 03:21 답글쓰기

김성엽 M 작성자

2022.11.21. 04:02 답글쓰기

황금잉어가물치 3

struct ValueList {int a, b, c;}; 선언한 후 struct ValueList values;로 구조체를 선언하는 예제를 보았습니다.
c언어처럼 struct valuelist {int a, b, c;} ValueList;를 선언 후 ValueList values;로 선언해도 되지 않는지 궁금합니다.

2022.12.25. 17:24 답글쓰기

김성엽 M 작성자

앞에 typedef를 사용하지 않으면 ValueList는 변수명일 뿐입니다. 따라서 ValueList는 변수명이기 때문에 다른 변수의 자료형을 사용할 수 없습니다.

2022.12.25. 17:59 답글쓰기

dh221009

댓글을 남겨보세요

등록