Journal of Engineering Education Research
Vol. 22, No. 1, pp. 29~38, January 2019

통합에 기반한 더 좋은 C 코드로의 변환 방안

김성기
한신대학교 컴퓨터공학부

A Transformation Method for Better C Code Based on Integration

Kim, Sung-ki
Division Of Computer Engineering, Hanshin University

ABSTRACT
Integration is an important intellectual ability to reconfigure several separated elements into one unified form and to concisely and categorically express them. In this paper, we classify the types of integration in C language, and propose a systematic and incremental method to transform to better code based on the integration types. This transformation method could also be used as a means to improve integrated thinking and efficiently learn C language, and will improve development ability in C programming or other language programming.

Keywords: Integration, Transformation, Statement, Function, Array, Structure, C language, Programming

I. 서 론

최근 4차 산업혁명에서 제조업과 정보통신기술을 융합해 작업 경쟁력을 높이는 것은 국가의 사활을 결정할 만큼 중요하다. 미래사회에 경쟁력을 기르기 위한 방법으로 소프트웨어 교육을 내세우며 세계 주요 국가에서는 컴퓨터 원리를 기반으로 문제를 해결해 내는 컴퓨팅 사고력 기반의 교육과정을 도입하고 있다(성정숙·김현철, 2015).

국내에서는 2015 개정 교육과정에서 정보 교과를 신설하여 초·중등학교에서 정보교과를 일정 이상 이수하게 하여 정보교육을 의무화하였고, 고등학교에서는 일반 선택으로 정보 교과를 이수할 수 있게 하였다(교육부, 2015; 성정숙·김현철, 2018).

대학교에서도 글로벌 경쟁력을 갖춘 소프트웨어 전문 인력과 소프트웨어 소양을 겸비한 융합인재를 양성하는 정책을 수립하고 추진하고 있다. 이에 따라, 대학들은 컴퓨팅 사고력을 핵심 역량으로 하는 소프트웨어 교육을 교양교육 과정에 채택하고 있다(김경민, 2017).

소프트웨어 교육은 소프트웨어 개발을 통하여 문제 해결 능력을 향상하는 코딩 교육을 포함한다. 코딩 교육은 프로그래밍 언어의 문법을 익히고 주어진 문제를 분석하여 필요한 자료와 처리과정을 통해 문제를 해결하는 프로그램 작성과정을 교육한다. C 프로그래밍을 배우는 과정에서 상수, 타입, 변수, 입출력문, 연산, 수식, 문장 등의 기초적인 요소들을 익힌 후 조건문, 반복문, 함수, 배열, 구조체, 포인터 등 본격적으로 프로그래밍에서 많이 사용되는 중요한 요소들을 배운다(Kernighan·Ritchie, 1988).

그런데, 코딩 교육 학습자들은 프로그래밍 언어 문법과 프로그래밍 과정에서 어려움을 겪는다(오경선·안성진, 2017; 최현종, 2011). 추상화 능력, 자동화 능력, 창의력, 융합 능력 등을 통하여 문제해결 능력을 향상시키는 컴퓨팅 사고력은 코딩에서의 어려움을 해결하는 방안으로 인식되고 있으며, 컴퓨팅 사고력을 향상하기 위한 방안이 국내외에서 활발히 연구되고 있다(김경민, 2017; 오경선·안성진, 2016; 최숙영, 2015). Kramer는 추상화 능력이 프로그래밍 능력에 기여한다고 주장하였다(Kramer, 2007).

통합은 개별적인 여러 요소들을 하나의 전체로 재구성하여 복잡한 양상을 단순하며 체계적인 형태로 표현한다(Dictionary; Techtarget). 프로그래밍 언어에도 여러 통합 관련된 요소들이 있다. C 언어의 경우, 여러 연산들로 구성되는 수식과 문장, 반복 수행될 문장들을 적절히 제어하여 반복 처리하는 반복문, 수행할 문장들을 간단히 함수이름으로 호출하는 함수, 여러 변수들을 하나의 변수로 구성하는 배열과 구조체 등은 통합의 원리가 내재된 요소들이다.

C 프로그래밍에서 통합 원리를 적용하여 코딩하는 방안은 아직 제안되지 않았으며, C 프로그래밍 교재(윤성우, 2010; 천인국, 2012)에서 통합의 개념, 통합이 적용된 예제, 통합의 결과와 이점 등에 대한 설명을 찾아보기 힘들다. 이로 인하여 C 학습자들이 초기 학습과정에서 이들을 통합적인 요소로서 이

Received November 15, 2018; Revised December 13, 2018
Accepted December 22, 2018
† Corresponding Author: skkim@hs.ac.kr

해하지 못하며, 여러 단계의 통합이 적용된 반복문과 함수를 배울 때에는 어려움을 경험한다. 이 어려움은 장기간의 많은 연습을 통하여 통합의 원리를 스스로 이해할 때까지 지속된다.

코드 변환은 동일한 기능을 수행하지만 보다 좋은 코드를 작성할 수 있게 한다. 소프트웨어 공학에서는 더 좋은 코딩에 대한 많은 제안이 이루어졌다. Sommerville(Sommerville, 2004)은 좋은 프로그램의 요건으로 유지보수성(maintainability), 신뢰성(dependability), 효율성(efficiency), 사용성(usability) 등 4 가지를 제안하였다. 또한 장기간동안 습득한 소프트웨어 품질 향상에 도움이 되는 비공식 코딩 규칙인 모범 코딩 프랙티스(best coding practice)가 소프트웨어 개발 커뮤니티에 의해 정립되었으며(Wikipedia), 주석 달기, 명명 관례, 간결한 코드, 변경 용이성, 가독성, 확장성, 모듈화, 코드와 데이터 분리 등이 제안되어 적용되고 있다.

본 논문에서는 C 언어의 주요 구성요소인 문장, 반복문 중의 for 문, 함수, 배열, 구조체 등에 내재한 통합 원리를 분석하여 통합 유형을 대체 통합, 동일화 통합, 묶음 통합, 매개변수화 통합 등으로 분류한다. 묶음 통합은 for 문으로의 묶음 통합, 함수로의 묶음 통합, 배열로의 묶음통합, 구조체로의 묶음 통합으로 세분화하여 분류한다.

또한 분류된 통합 유형에 따라 초보적 수준의 C 코드를 더 좋은 코드로 변환하는 방안을 제시한다. 제시된 변환 방안은 초보자 수준의 코드를 고수준의 코드로 점진적이며 단계별로 변환하는 방법으로, 통합의 원리를 이해하면 쉽게 C 코딩에서 활용할 수 있는 방법이다.

제안된 변환 방안은 C 프로그래밍 외에 Java나 Python 등의 다른 프로그래밍에서도 적용가능하며, 통합 기반의 코드 변환 방법은 코딩 능력을 빠르게 향상시킬 수 있을 것이다.

II. 통합의 정의와 분류

1. 통합의 정의

Dictionary.com에서는 통합을 “Integration is an act or instance of combining into an integral whole”로 정의하고(Dictionary), Techtarget.com에서는 이와 유사하게 “Integration is the act of bringing together smaller components into a single system that functions as one”으로 정의한다(Techtarget).

이는 서로 다른 것에서 공통된 성질을 파악하여 통일된 형태로 표현하며, 여러 요소들을 하나의 전체로서 기능하게 한다. 그 결과 분리되었던 각 개별적 요소가 전체의 한 부분의 역할을 하며, 개별적인 요소를 접근하려면 전체의 한 요소로 접근된다. 통합은 다음과 같이 정의된다.

[정의 1] 통합은 여러 통합 대상들 o_1, o_2, ... o_n에서 통합 결과물 I로의 사상(mapping)으로, 통합 대상들 o_1, o_2, ... o_n을 통합의 내포(intention), 통합 결과물 I를 통합의 외연(extension)이라 한다.

통합이 중요한 역할을 하는 학문분야가 수학이다. 수학은 개별적인 객체 하나하나가 아니라 객체 전체에 대한 기본 원리 또는 공식을 구하는 학문이므로 수학은 객체들의 통합을 추구하며, 통합적 사고가 수학의 밑바탕이 된다.

수학에서 통합의 대표적 예는 집합 원소들을 나열하는 원소 나열법을 모든 원소들이 만족하는 조건으로 표시하는 조건제시법으로 변환하는 것이다.

$$S = \{\ 3, 5, 6, 9, 10, 12, 15, 18, 20, 21, 24, 25, 27, 30\ \}$$
$$== \text{(통합)} ==>$$
$$S = \{\ x \mid 3 \le x \le 30 \ \wedge (\ x\text{는 3의 배수} \ \vee\ x\text{는 5의 배수}\ )\ \}$$

또한 좌표 상에 나열되는 무한한 점들을 방정식으로 간단히 표현하는 과정과 수열의 일반항을 구하는 과정도 통합의 예이다.

2. C 언어에서의 통합 분류

본 장에서는 C 언어에 나타난 통합 유형을 분류한다. 통합은 크게 대체 통합, 동일화 통합, 묶음 통합, 매개변수화 통합으로 분류되며, 묶음 통합은 다시 for 문으로의 묶음 통합, 함수로의 묶음 통합, 배열로의 묶음 통합, 구조체로의 묶음 통합으로 분류된다. Table 1은 통합 전후의 요소, 통합 이전과 이후의 양상, 양상에 대한 설명, 장점을 보여준다.

가. 대체 통합

대체 통합(substituting integration)은 한 요소를 동일한 다른 요소로 대체하여 분리되었던 요소들을 하나로 변환하는 통합으로, 수식 e를 변수 v에 대입하는 대입문 ≪v = e;≫가 주어졌을 때, 변수 v를 포함하는 문장 ≪s(v);≫에서의 v를 e 또는 (v = e)로 대체하여 하나의 문장 ≪s(v ↦ e);≫ 또는 ≪s(v ↦ (v = e));≫로 변환한다.

대체 통합을 수행하는 과정은 다음과 같다.

과정 1: ≪v = e;≫ 문장의 변수 v가 포함된 문장을 찾는다.
과정 2: 문장의 v를 e 또는 ≪v = e≫로 대체한다. v를 e로 대체할 때에는 모든 v에 대해, v를 ≪v = e≫로 대체할 때는 첫 번째 v에 대해서만 대체한다.
과정 3: ≪v = e;≫ 문장을 삭제한다.

Table 1 Classification and Aspects of Integration Components in C Language

통합 유형	통합 전 요소	통합 이전의 양상	통합 후 요소	통합 이후의 양상	통합 양상 설명	장점
대체 통합	문장들	v = e; s(v);	한 문장	s(v ↦ e); 또는 s(v ↦ (v = e));	변수 v를 사용하는 문장 s(v);에서 v를 대입문 v = e;의 e 또는 v = e로 대체	간결한 코드 및 가독성 증대
동일화 통합	유사한 문장들	s1; s2; ... sn;	동일한 문장들	s0; s'; s"; ... , s'; s";	문장 s1;, s2;, ... sn;을 문장 s0;와 동일한 형태의 문장들 s'; s";로 변환	for 문으로의 묶음 통합을 가능하게 함
for 문으로의 묶음 통합	동일 문장들	s; s; ... si;	for 문	for (i=0; i < n; i++) s'i	n번 나열되는 동일한 문장들 s;를 for 문으로 통합	간결한 코드 및 반복회수 파악 및 반복회수 변경 용이성 증대
함수로의 묶음 통합	여러 문장들	s1; s2; ... sn;	함수	f() { s1; s2; ... sn; }	여러 문장 s1;, s2;, ... sn;을 함수로 통합	프로그램의 기능적 구성, 가독성 및 사용성 증대
배열로의 묶음 통합	여러 변수	t v1, v2, ... vn;	배열	t v[n];	동일한 타입 t 변수들을 배열로 통합	하나의 단위로 관리 가능 및 for 문과 결합 가능, 함수와 결합 가능
구조체로의 묶음 통합	여러 변수	t1 v1; ... tn vn;	구조체	struct {t1 v1; ... tn vn; } v;	이질적 타입 t1, t2, ... tn 변수들을 구조체로 통합	연관된 데이터를 하나의 단위로 관리 가능, 함수와 결합 가능
매개변수화 통합	여러 유사한 함수	f_d1(), f_d2(), ... f_dn()	매개변수 갖는 함수	f(t v)	타입 t의 값들 d1, ... dn이 함수이름에 포함된 함수들을 매개변수 갖는 함수로 통합	함수 개수의 감소, for 문과 결합 가능, 생산성 및 사용성 증대

Fig. 1은 대체 통합의 예를 보여준다. 변수 a와 b가 사용된 문장에서 a와 b를 대입문에서의 오른쪽 수식으로 대체하였다.

```
a = cos(x)*cos(x);
b = 1 – a;
sinx = sqrt(b)
```

⇨

```
sinx = sqrt(1–cos(x)*cos(x));
```

Fig. 1 Substituting Integration to a Single Statement

Fig. 2는 대체 통합의 또 다른 예를 보여준다.

```
c = getchar();
if (c>='0' && c<='9')
   n = c - '0';
```

⇨

```
if ((c=getchar())>='0'
                  && c<='9')
n = c - '0';
```

Fig. 2 Substituting Integration to a Single Statement

여기에서는 if 문에서 사용된 첫 번째 변수 c를 대입문 ≪c = getchar();≫에서의 (c = getchar())로 대체하여 2 문장을 if 문 하나로 통합하였다.

나. 동일화 통합

동일화 통합(unifying integration)은 여러 유사한 요소들의 공통점을 파악하여 동일한 형태로 변환하는 통합으로, 유사한 요소들이란 여러 요소들이 완전히 동일하지 않지만 일정한 변화 규칙을 가지면서 바뀌는 요소들을 의미한다. 유사한 문장들 ≪s1;, s2;, ... sn;≫이 주어졌을 때, 동일화 통합은 문장 ≪si;≫를 ≪s'; s";≫로 변환하여 전체를 ≪s0; s'; s"; ... s'; s";≫의 동일한 형태의 문장들로 재구성한다.

동일화 통합을 수행하는 과정은 다음과 같다.

과정 1: 유사한 문장들 ≪s_1;, s_2;, ... s_n;≫에서 동일한 부분과 변화하는 부분을 파악한다.

과정 2: 변화하는 부분의 변화규칙을 파악하여 이를 변수 v로 표현하는 수식 s"을 구성한다.

과정 3: 문장들 ≪s_1;, s_2;, ... s_n;≫을 변수 v를 이용하여 동일한 문장 ≪s';≫로 재구성하고 문장 ≪s";≫를 재구성된 문장 다음에 배치한다.

과정 4: v의 초기값을 설정하는 문장 ≪s_0;≫을 작성하여 처음에 배치한다.

Fig. 3은 동일화 통합의 예를 보여준다.

```
int sum = 0;
sum = sum + 2;
sum = sum + 5;
sum = sum + 8;
sum = sum + 11;
```

⇨

```
int sum = 0;
int i = 2;
sum = sum + i;  i += 3;
sum = sum + i;  i += 3;
sum = sum + i;  i += 3;
sum = sum + i;  i += 3;
```

Fig. 3 Unifying Integration to Same Statements

여기에서는 유사한 문장들에서 규칙적으로 3씩 증가하는 3, 5, 8, 11을 변수 i로 대체하고 문장 ≪i += 3;≫을 각 문장 뒤에 추가한 후 i를 초기화하는 문장 ≪i = 2;≫를 처음에 배치하여 유사한 4 문장들을 완전히 동일한 문장들로 통합하였다.

다. 묶음 통합

묶음 통합(binding integration)은 여러 요소들을 하나의 전체로 묶어 분리되었던 개별 요소들을 전체적인 하나로 변환하는 통합으로, ≪c1, c2, ... cn≫을 하나의 통합된 요소 c로 변환한다.

묶음 통합은 for 문으로의 묶음 통합, 함수로의 묶음 통합, 배열로의 묶음 통합, 구조체로의 묶음 통합으로 나누어진다.

for 문으로의 묶음 통합을 수행하는 과정은 다음과 같다.

과정 1: 동일화 통합이 이루어진 문장들 ≪s_0; s; ... s;≫이 주어졌을 때, 문장들에서 초기값 설정 문장 ≪s_0;≫를 확인한다.
과정 2: 동일한 문장들의 개수를 세어 for 문에서의 반복회수n을 구한다.
과정 3: 제어변수 cv를 이용하여 다음의 for 문으로 변환한다.
for (s0, cv = 0; cv < n; cv++) s;
과정 4: s0;와 s;에 cv를 대체할 수 있는 변수가 있으면 이를 이용하여 for 문을 간소화한다.

함수로의 묶음 통합, 배열로의 묶음 통합, 구조체로의 묶음 통합을 수행하는 과정은 다른 통합과정에 비해 간단하여 Table 1의 통합 이전 양상과 통합 이후의 양상으로 파악할 수 있으므로 생략한다.

Fig. 4는 for 문으로의 묶음 통합의 예를 보여주며, 동일한 4개의 문장들을 for 문으로 통합하였다. 특히, 처음 통합된 for 문에서의 제어변수 cv 대신 제어변수 i를 사용하여 for 문을 간소화하였다.

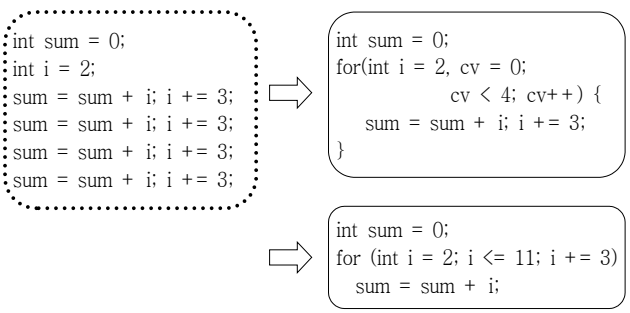

Fig. 4 Binding Integration to a for Statement

Fig. 5는 함수로의 묶음 통합의 예를 보여주며, for 문과 putchar() 호출문을 함수로 통합하였다.

Fig. 6은 배열로의 묶음 통합의 예를 보여주며, 동일한 타입의 여러 변수들을 배열로 통합하였다.

Fig. 7은 구조체로의 묶음 통합의 예를 보여며, 서로 다른 타입의 여러 변수들을 구조체 변수 person으로 통합하였다.

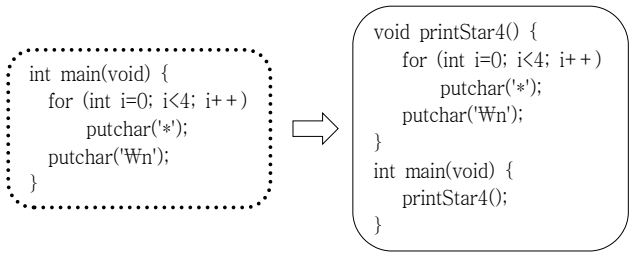

Fig. 5 Binding Integration to a Function

Fig. 6 Binding Integration to an Array

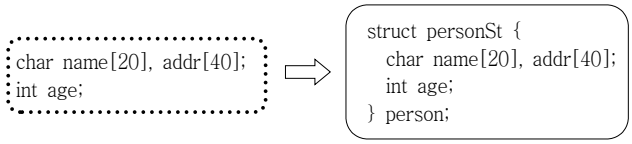

Fig. 7 Binding Integration to a Structure

라. 매개변수화 통합

매개변수화 통합(parameterizing integration)은 여러 유사한 함수들의 공통점을 파악하여 이를 매개변수를 갖는 하나의 함수로 변환하는 통합으로, ≪ f_d1() { ... }, f_d2(){ ... }, ... f_dn(){ ... }≫에서와 같이 함수이름에 타입 t의 값들 d1, ... dn이 포함된 함수들을 매개변수를 갖는 하나의 함수 ≪f(t v){ ... }≫로 재구성한다.

매개변수화 통합을 수행하는 과정은 다음과 같다.

과정 1: 함수명에 데이터가 표시된 여러 함수들 ≪f_d_1() { ... }, f_d_2(){ ... }, ... f_d_n(){ ... }≫에 나타난 데이터 d_1, d_2 ... d_n의 타입 t를 결정한다.
과정 2: 함수의 시그니처를 f(t v)로 구성한다.
과정 3: f_d_1()의 구현부분을 d_1이 아니라 형식 매개변수 v에 대한 구현으로 변경하여 f(t v)의 구현을 완성한다.

Fig. 8은 매개변수화 통합의 예를 보여주며, 다른 개수의 *를 출력하는 여러 유사한 함수들을 매개변수 n을 갖는 하나의 함수로 통합하였다.

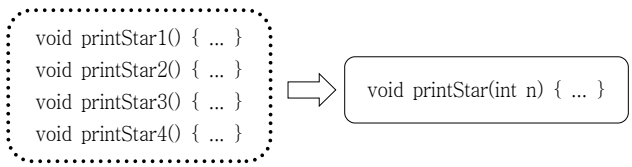

Fig. 8 Parameterizing Integration of Similar Functions

3. 통합과 추상화의 관계

추상화는 사물을 이해하고 다루기 쉽게 단순화시키는 작업으로, 사물과 관련된 구체적인 사항은 최대한 감추거나 생략하고 핵심부분만 분리해내어 구체적 사물에 대응된 추상물을 생성한다(김성기, 2004; Kramer, 2007). 김성기는 추상화의 여러 양상을 비교분석하여 추상화 유형을 분류하였다(김성기, 2004).

본 논문에서의 통합은 추상화와 밀접한 관련성을 가진다. 예를 들어, 배열로의 묶음 통합과 구조체로의 묶음 통합은 데이터 추상화와 대응되며, 함수로의 묶음 통합은 절차적 추상화에 대응된다.

한편, 대체 통합, 동일화 통합, for 문으로의 묶음 통합에 대응되는 추상화는 없다. 본 논문에서는 함수로의 묶음 통합과 매개변수화 통합을 다른 유형의 통합으로 분류하지만 추상화에서는 매개변수를 갖는 함수도 절차적 추상화에 포함되므로 매개변수화 통합도 절차적 추상화에 대응된다.

III. 통합 유형에 기반한 더 좋은 코드로의 변환

본 장에서는 2장에서 살펴본 통합 유형에 따라 C 코드를 더 좋은 코드로 변환하는 방안을 제시한다. 통합 유형에 따라 초보자 수준의 단순 코드를 고수준의 좋은 코드로 변환하는 체계적 과정을 통하여 통합의 원리를 이해하고 좋은 코드의 요건과 장점도 확인할 수 있다.

1. 대체 통합으로 간결한 코드로의 변환

대체 통합은 대입문과 대입문의 결과를 사용하는 문장들을 하나의 문장으로 변환하며, 그 결과 코드가 간결해지며 가독성이 높아진다.

Fig 1과 Fig. 2에서는 분리되었던 문장들에 대해 대체 통합을 수행하여 간결한 문장으로 변환하였다.

다음 코드는 연속된 디지트 문자들을 getchar() 함수로 입력하여 int 타입의 정수로 구성한다.

```
int c, n = 0;
c = getchar();
while (c >= '0' && c <= '9') {
    n = n * 10 + c - '0';
    c = getchar();
}
```

≪c = getchar();≫이 2번 중복 작성된 이 코드를 대체 통합하여 변환한 코드는 다음과 같다.

```
int c, n = 0;
while ((c = getchar()) >= '0' && c <= '9')
    n = n * 10 + c - '0';
```

이 코드는 이전 코드에 비해 간결하며 가독성이 우수하며, 중복이 제거되어 변경 용이성이 좋다.

2. 동일화 통합과 for 문으로의 묶음 통합으로 반복 문장들을 for 문으로 변환하기

가. 동일한 반복문장들을 for 문으로 변환하기

동일한 반복문장들이 나열되면 묶음 통합을 통하여 for 문이나 while 문으로 쉽게 변환할 수 있다.

다음 코드는 *를 4번 출력하며, 동일한 문장 ≪putchar('*');≫가 4번 반복되었다.

```
putchar('*');   putchar('*');   putchar('*');   putchar('*');
```

이 코드의 반복회수는 4인데, 만약 반복 문장들이 20개 정도이면 반복 회수 파악도 쉽지 않으며, 반복 회수를 100번으로 변경하는 것도 어렵다.

동일한 4 문장들에 대해 묶음 통합을 수행하여 for 문으로 변환한 코드는 다음과 같다.

```
for (int i = 0; i < 4; i++)
    putchar('*');
```

for 문으로 묶음 통합된 이 코드는 이전 코드보다 반복회수 파악과 반복회수 변경이 훨씬 용이하므로 가독성과 확장성이 좋다.

나. 유사한 반복문장들을 for 문으로 변환하기

유사한 반복문장들이 나열된 경우 동일화 통합과 묶음 통합으로 for 문으로 변환할 수 있다. 다음 코드는 2, 5, 8, 11의 합을 구한다.

```
int sum = 0;
sum = sum + 2;    sum = sum + 5;
sum = sum + 8;    sum = sum + 11;
```

2부터 11까지 3씩 증가하는 정수의 합을 구하는 이들 유사한 4 문장들에 대해 동일화 통합을 수행하면 다음과 같이 완전히 동일한 2 문장 단위의 4 반복부분으로 변환된다. 이 때 변화하는 부분을 위한 변수 i의 초기화 부분이 추가된다.

```
int sum = 0;
int i = 2;
sum = sum + i; i += 3;    sum = sum + i; i += 3;
sum = sum + i; i += 3;    sum = sum + i; i += 3;
```

동일한 반복부분들로 변환된 이 코드에 대해 for 문으로의 묶음 통합을 수행하면 다음의 for 문으로 변환된다.

```
int sum = 0;
for (int i = 2; i <= 11; i += 3)
    sum = sum + i;
```

이 코드는 반복되는 상황을 쉽게 파악할 수 있고, for 문에서 반복 조건을 쉽게 변경할 수 있으므로 가독성, 변경 용이성이 이전 코드보다 우수하다.

어느 정도 숙달되면 유사한 반복문장들을 동일한 반복부분들로 일일이 변환하지 않고서도 바로 for 문으로 변환할 수 있다. 그 때까지 많은 연습을 통하여 for 문에 내재한 동일화 통합과 묶음 통합의 원리를 이해해야 한다.

다. 복잡한 반복문장들을 for 문으로 변환하기

유사하면서 다소 복잡한 반복문장들의 경우 for 문으로의 변환이 쉽지 않다. 그 이유는 복잡한 반복문장들에 대한 동일화 통합이 쉽지 않기 때문이다. 이 경우 동일화 통합 과정을 명시적으로 수행한 후 묶음 통합을 수행하여 for 문으로 변환하는 것이 효율적이다.

다음 코드는 밑변이 * 3개로 구성되는 직각삼각형을 그린다.

```
for (int i = 0; i < 3; i++) putchar(' ');   // 반복
for (int i = 0; i < 1; i++) putchar('*');   // 부분
putchar('₩n');                              // 1

for (int i = 0; i < 2; i++) putchar(' ');   // 반복
for (int i = 0; i < 2; i++) putchar('*');   // 부분
putchar('₩n');                              // 2

for (int i = 0; i < 1; i++) putchar(' ');   // 반복
for (int i = 0; i < 3; i++) putchar('*');   // 부분
putchar('₩n');                              // 3
```

```
*
**
***
```

이 코드에서 각 라인에 출력되는 공백문자 개수는 하나씩 감소하며, * 개수는 하나씩 증가한다. 그 결과 3 문장 단위로 3번 반복된다.

for 문으로 변환하기 위해서는 유사한 반복부분들을 모두 동일한 형태로 변환하는 동일화 통합을 수행해야 한다. 3 문장 단위의 3 반복부분들을 완전히 동일한 반복부분으로 변환하도록 동일화 통합을 수행한 결과는 다음과 같다.

```
int cntBlank = 3, cntStar = 1;

for (int i = 0; i < cntBlank; i++) putchar(' ');
for (int i = 0; i < cntStar; i++) putchar('*');
putchar('₩n');
cntBlank--; cntStar++;

for (int i = 0; i < cntBlank; i++) putchar(' ');
for (int i = 0; i < cntStar; i++) putchar('*');
putchar('₩n');
cntBlank--; cntStar++;

for (int i = 0; i < cntBlank; i++) putchar(' ');
for (int i = 0; i < cntStar; i++) putchar('*');
putchar('₩n');
cntBlank--; cntStar++;
```

변환된 이들 5 문장 단위의 3 반복부분들은 모두 동일한 문장들로 통합되었으므로 묶음 통합을 수행하면 다음의 for 문으로 변환된다.

```
for (int cntBlank = 3, cntStar = 1; cntStar <= 3;
                                cntBlank--, cntStar++) {
    for (int i = 0; i < cntBlank; i++) putchar(' ');
    for (int i = 0; i < cntStar; i++) putchar('*');
    putchar('₩n');
}
```

복잡한 for 문 작성과정에는 동일화 통합이 필수적으로 적용된다. 그러므로 for 문 학습과정에는 동일화 통합 원리를 확실한 이해해야 하며, 이를 위한 다양한 예제들을 통한 많은 연습이 요구된다.

3. 함수에서의 묶음 통합, 매개변수화 통합, 동일화 통합

C 언어의 함수는 여러 문장들을 기능을 나타내는 이름으로 추상화 것이다(Kernighan·Ritchie, 1988; Wikipedia). 한편, 통합의 관점에서 함수는 여러 문장들을 묶음 통합시킨 것이다.

C 함수는 C 프로그래밍 교육에서 가장 어려운 부분 중 하나인데, 그 이유는 통합의 관점에서 함수는 함수로의 묶음 통합과 매개변수화 통합이 내재되고 다시 동일화 통합을 통하여 for 문으로 묶음 통합되어지는 고수준의 통합 산물이기 때문이다.

가. 묶음 통합으로 문장들을 함수로 변환하기

다음 코드는 for 문을 이용하여 *를 4개 출력한 후 줄바꿈 문자를 출력한다.

```
for (int i = 0; i < 4; i++)
    putchar('*');
putchar('₩n');
```

한 라인에 * 4개를 출력하는 기능을 가지는 이 코드에 대해 함수로의 묶음 통합을 수행하면 함수명이 printStar4인 함수가 작성되고 main()에서 호출된다.

```
void printStar4() {   // 한 라인에 '*' 4개 출력하는 함수
    for (int i = 0; i < 4; i++)
        putchar('*');
    putchar('₩n');
}
int main(void) {
    printStar4();   // 한 라인에 '*' 4개 출력하는 함수 호출
}
```

이 코드와 같이 여러 문장들을 함수로 구성하면 프로그램이 모듈화되고 기능적으로 작성되어 가독성, 변경 용이성, 응집도 등이 좋아진다.

나. 매개변수 갖는 함수로 변환하기

함수 printStar4()를 이용하여 한 라인에 * 4개를 출력할 수 있지만, 한 라인에 * 2개 또는 3개를 출력할 수 없다. 그 결과, 함수를 이용하여 * 개수를 다르게 출력하려면 새로운 함수를 작성해야 한며. 작성해야할 함수 개수가 많아진다.

다음 코드는 한 라인에 특정 개수의 *를 출력하는 함수들을 호출하여 라인마다 *를 1개부터 4개까지 출력한다.

```
void printStar1() { ... } // 한 라인에 * 1개 출력하는 함수
void printStar2() { ... } // 한 라인에 * 2개 출력하는 함수
void printStar3() { ... } // 한 라인에 * 3개 출력하는 함수
void printStar4() {       // 한 라인에 * 4개 출력하는 함수
    for (int i = 0; i < 4; i++)
        putchar('*');
    putchar('₩n');
}
int main(void) {
    printStar1();   printStar2();
    printStar3();   printStar4();
}
```

```
*
**
***
****
```

printStar1() 함수부터 printStar4()까지 4개 함수는 출력할 * 개수를 함수이름에 포함하는데, * 출력 개수에서만 차이를 갖는 유사한 기능의 함수이다.

유사한 기능의 여러 함수들에 대해 매개변수화 통합을 수행하면 하나의 함수로 재작성할 수 있으며, 이때 매개변수(parameter)가 새로이 도입된다(Wikipedia).

printStar1(), printStar2(), printStar3(), printStar4()의 경우 함수이름에 표시된 출력 개수 1, 2, 3, 4를 정수 매개변수 n으로 대체한 printStar(int n)로 매개변수화 통합된다. 그 결과, 통합된 함수 printStar(int n)는 실 매개변수에 주어지는 개수의 *를 출력할 수 있다.

다음 코드는 매개변수화 통합으로 작성된 함수 printStar(int n)를 이용하여 라인마다 *를 1개부터 4개까지 출력한다.

```
void printStar(int n) {   // 매개변수를 갖는 통합된 함수
    for (int i = 0; i < n; i++)
        putchar('*');
    putchar('₩n');
}
int main(void) {
    printStar(1);   printStar(2);   printStar(3);   printStar(4);
}
```

함수로의 매개변수화 통합은 작성해야할 함수의 개수를 크게 줄이는 효과가 있으므로 프로그래머의 생산성을 크게 증가시키며 사용성을 높인다.

이 코드에서는 매개변수를 갖는 함수가 반복적으로 호출되었는데, 또다시 동일화 통합과 묶음 통합을 수행하여 for 문으로 변환할 수 있다. 위의 코드에 대해 for 문으로 변환한 결과는 다음과 같다.

```
void printStar(int n) {
    for (int i = 0; i < n; i++)
        putchar('*');
    putchar('₩n');
}
int main(void) {
    for (int i = 0; i < 4; i++)
        printStar(i);
}
```

이 코드는 이전 코드들과 동일한 결과를 출력하지만 간결하면서 변경 용이성이 훨씬 우수하다. 이는 반복 문장들이 for 문으로 통합될 때의 이점과 동일하다.

다. 매개변수화 통합된 함수의 매개변수화 통합

printStar(int n) 함수는 한 라인에 *를 n개 출력한다. 그런데, 출력할 문자가 $, A 또는 B인 경우 이를 위한 새로운 함수를 또다시 작성해야 한다.

다음 코드는 한 라인에 *, $, A, B를 출력하는 함수를 이용하여 라인마다 *, $, A, B를 1개부터 4개까지 출력한다.

```
void printStar(int n) { ... }    // *를 n번 출력
void printDollor(int n) { ... } // $를 n번 출력
void printA(int n) { ... }       // A를 n번 출력
void printB(int n) {             // B를 n번 출력
    for (int i = 0; i < n; i++)
        putchar('B');
    putchar('₩n');
}
int main(void) {
    printStar(1); printDollor(2);
    printA(3);    printB(4);
}
```

```
*
$$
AAA
BBBB
```

printStar(int n), printDollor(int n), printA(int n), printB(int n)도 출력할 문자만 다른 유사한 기능의 함수이며, 함수이름에 출력할 문자 정보를 가진다.

이들 함수들은 또다시 매개변수화 통합이 가능하며, 그 결과 함수이름에 표시된 출력 문자를 새로운 매개변수로 대치한 함수 printChar(char c, int n)가 작성된다.

다음 코드는 2번 매개변수화 통합된 함수printChar(char c, int n)를 이용하여 라인마다 *, $, A, B를 1개부터 4개까지 차례로 출력한다.

```
void printChar(char c, int n) {
    for (int i = 0; i < n; i++)
        putchar(c);
    putchar('₩n');
}
int main(void) {
    printChar('*', 1);     printChar('$', 2);
    printChar('A', 3);     printChar('B', 4);
}
```

```
*
$$
AAA
BBBB
```

2개의 매개변수를 갖는 printChar(char c, int n) 함수는 한 라인에 임의 문자를 임의 개수만큼 출력할 수 있는 강력한 기능의 함수인데, 묶음 통합과 2번의 매개변수화 통합이 내재된 고차원적 통합 함수이다.

이 코드의 main() 함수의 함수 호출문들도 배열을 활용하여 for 문으로 묶음 통합될 수 있다.

C 프로그래밍의 전문가 수준에 도달하면 묶음 통합과 매개변수화 통합을 명시적으로 수행하지 않고서도 여러 매개변수를 가지는 함수를 자유로이 작성할 수 있다. 그러나 초보자는 함수로의 묶음 통합과 매개변수화 통합, for 문에서의 동일화 통합과 묶음 통합을 수행하는 여러 예제와 연습을 통해 함수의 기본 원리인 추상화, 매개변수의 원리, 반복문에서의 함수의 반복 호출 등을 이해할 필요가 있다.

4. 배열에서의 묶음 통합, 동일화 통합, 매개변수화 통합

가. 배열의 묶음 통합에서 매개변수화 통합까지

단순변수 4개를 사용하여 4개 점수를 입력하여 저장하는 C 코드는 다음과 같다.

```
int score1, score2, score3, score4;
scanf("%d", &score1); scanf("%d", &score2);
scanf("%d", &score3); scanf("%d", &score4);
```

점수 개수가 4개인 경우는 변수 선언과 입력이 간단하지만 만약 점수 개수가 100개이면 이 스타일의 코드로 작성하는 것은 쉽지 않다.

동일한 타입의 여러 연관된 데이터는 배열로의 묶음 통합을 수행하여 배열로 변환할 수 있으며, 다음 코드는 단순변수 4개를 scores 배열로 변환하였다.

```
int main(void) {
    int scores[4];
    scanf("%d", &score[0]); scanf("%d", &score[1]);
    scanf("%d", &score[2]); scanf("%d", &score[3]);
}
```

변환된 코드에서 scores 배열의 인덱스가 0부터 3까지 규칙적으로 변하므로 4개의 유사한 scanf() 함수 호출문에 대해 동일화 통합을 수행할 수 있으며, 다시 for 문으로의 묶음 통합을 수행하여 for 문으로 변환한 코드는 다음과 같다.

```
int main(void) {
    int scores[4];
    for (int i = 0; i < 4; i++)
        scanf("%d", &scores[i]);
}
```

변환된 for 문은 4개의 정수를 입력하는 기능을 가지므로 함수로의 묶음 통합을 수행한 코드는 다음과 같다.

```
int scores[4];                    // 전역변수로 선언한 배열
void input4IntsToScores() {  // 정수 4개 입력하는 함수
    for (int i = 0; i < 4; i++)
        scanf("%d", &scores[i]);
}
int main(void) {
    input4IntsToScores();       // scores에 정수 4개 입력
}
```

여기에서의 함수 input4IntsToScores()은 매개변수가 없는 함수로서, 전역변수 scores 배열에 4개의 원소를 입력한다.

이 함수에 대해 매개변수화 통합을 수행하면 입력 개수와 입력 배열을 매개변수를 갖는 함수 inputInts(int n, int arr[])로 변환된다.

```
#define NOSCORES 4
void inputInts(int n, int arr[]) {
    for (int i = 0; i < n; i++)
      scanf("%d", &arr[i]);
}
int main(void) {
    int scores[NOSCORES];
    inputInts(NOSCORES, scores);
}
```

이 코드는 이전의 코드에 비해 모듈화가 잘 되었고 저장 개수와 저장 배열을 쉽게 변경할 수 있으므로 변경 용이성도 우수하다.

여러 데이터를 배열에 저장하고 배열을 처리하는 함수 작성 및 호출 과정에는 모든 유형의 통합이 적용된 것을 확인하였다. 이것은 C 프로그래밍 학습에서 함수와 연관된 배열 부분을 제대로 학습하기 어려운 이유를 단적으로 보여주는 것이다.

이러한 어려움을 극복하기 위해서는 묶음 통합, 동일화 통합, 매개변수화 통합 등 배열에 내재된 통합 원리를 이해하고 활용하는 수준이 될 수 있도록 연습해야 할 것이다.

나. 배열을 이용한 데이터와 처리과정의 분리

다음 코드는 은행 업무를 나타내는 4개 메뉴를 출력하기 위해 4개 문자열을 printf() 문에서 사용한다.

```
printf("1 open account₩n");   printf("2. deposit₩n");
printf("3. withdraw₩n");      printf("4. transfer₩n");
```

이 코드는 4개의 문자열이 printf() 함수에서 사용되므로 메뉴 순서 변경, 새로운 메뉴 첨가, 메뉴 삭제 등의 작업을 수행하려면 여러 부분이 영향을 받는다. 그러므로 이 코드는 좋은 코드가 아니다.

printf() 함수 내의 4개 문자열 배열로 묶음 통합하면 다음과 같이 데이터와 처리과정이 분리되어 for 문으로 변경 가능한 더 좋은 코드로 변환된다.

```
char *menus[] = { "open Account", "deposit",
                  "withdraw",     "transfer" };
printf("1. %s₩n", menus[0]);  printf("2. %s₩n", menus[1]);
printf("3. %s₩n", menus[2]);  printf("4. %s₩n", menus[3]);
```

이 코드에 대해 동일화 통합과 for 문으로의 묶음 통합을 수행하면 다음의 for 문으로 변환된다.

```
char *menus[] = { "open Account", "deposit",
                  "withdraw",     "transfer" };
for (int i = 0; i < sizeof(menus) / sizeof(char *); i++)
    printf("%d. %s\n", (i+1),   menus[i]);
```

이 코드를 메뉴를 출력하는 함수로 구성하기 위해 함수로의 묶음 통합과 매개변수화 통합을 수행한 코드는 다음과 같다.

```
void printMenus(char *menus[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d. %s₩n", (i+1),   menus[i]);
}
char *menus[] = { "open Account", "deposit",
                  "withdraw",     "transfer" };
int main(void) {
    printMenus(menus, sizeof(menus) / sizeof(char *));
}
```

이 코드는 메뉴의 순서 변경, 메뉴 추가 및 삭제 등이 용이하므로 변경 용이성이 우수하다. 이러한 이점이 제공되는 이유는 배열을 이용하여 처리코드와 데이터를 분리하였기 때문이며, 이 과정에서도 통합에 기반한 코드 변환이 활용되었다.

5. 구조체에서의 묶음 통합, 동일화 통합, 매개변수화 통합

묶음 통합의 결과물인 구조체는 연관된 변수들을 하나의 단위로 다룰 수 있으며, 구조체 전체를 함수 매개변수로 전달 가능한 편리한 요소이다.

연관된 정보인 이름, 주소, 나이를 분리된 별도의 변수들로 사용하는 코드는 다음과 같다.

```
char name[20],  addr[40];  int age;
scanf("%s %s %d", name, addr, &age);
```

이름, 주소, 나이에 대해 구조체로의 묶음 통합을 수행하면 다음의 코드로 변환된다.

```
struct personSt {              // 구조체 선언
  char name[20],    addr[40];    int age;
}
int main(void) {
  struct personSt person;  // 구조체 변수 선언
  scanf("%s %s %d", person.name, person.addr,
                    &person.age);
}
```

personSt 구조체 필드에 값을 입력하는 코드에 대해 함수로의 묶음 통합을 수행하면 다음 코드로 변환된다.

```
struct personSt {              // 구조체 선언
    char name[20],    addr[40];    int age;
};
void inputPerson(struct personSt *p) { // 필드 입력 함수
    scanf("%s %s %d", p->name, p->addr, &p->age);
}
int main(void) {
    struct personSt person; // 구조체 변수 선언
    inputPerson(&person);  // 구조체 필드 입력 함수 호출
}
```

구조체에서의 통합 난이도는 배열에 비하여 낮으므로 배열에서의 통합을 이해하면 구조체에서의 통합은 쉽게 이해될 수 있다.

6. 변환 방안의 활용

제안된 통합에 기반한 좋은 코드로의 변환 방안은 통합이라는 체계적인 사고과정에 기반하여 저수준 코드에서 고수준 코드로의 단계적이며 점진적인 이행을 포함한다. 그러므로 이 변환 방안은 C 프로그램의 중요 요소들의 기본 원리, 특징, 장단점을 쉽게 이해할 수 있게 한다. 또한 통합적 사고를 향상시키는 수단과 C 프로그래밍을 효율적으로 학습하기 위한 수단으로도 사용될 수 있다.

현재 필자는 제안된 변환 방안을 C 언어와 Java 프로그래밍 교육에 활용하고 있으며, 특히 for 문과 함수를 단계적이며 체계적으로 학습시킨 결과 이해도가 높아지고 코드 작성 능력이 배양된 것을 관찰하였다. 보다 객관적인 효용성을 확인하기 위해서는 제시된 변환 방안을 활용하여 통합적 사고와 프로그래밍을 효율적으로 교육할 수 있는 수업계획안을 설계하고 실행 및 평가를 수행하는 연구가 추가적으로 필요하다.

IV. 결 론

본 논문에서는 C 언어의 수식과 대입문, for 문, 함수, 배열, 구조체 등 중요 요소들에 내포된 통합 원리를 분석하여 통합 유형을 분류하였다. 또한 분류된 통합 유형에 기반하여 이들 요소가 포함된 코드를 보다 좋은 코드로 변환하는 방안을 제시하였다. 제시된 변환 방안은 통합 원리를 보다 빠르게 이해할 수 있게 하므로 통합적 사고를 향상시키는 도구와 C 언어 학습에서 핵심적인 요소들을 효율적이며 체계적으로 학습하는 도구로도 사용될 수 있을 것이다.

추후 연구과제는 다음의 2가지이다. 첫째, 컴퓨팅 사고에서는 추상화 능력, 자동화 능력, 창의력, 융합능력이 문제 해결 능력을 향상시키는 요인으로 강조된다. 이와 더불어 통합 능력도 문제해결 능력을 향상하는 중요한 요인임을 검증하고 통합 능력을 향상할 수 있는 방안을 강구하는 것이다. 둘째, 제시된 변환 방안을 적용하는 수업계획안을 설계하고 실행 및 평가를 위한 연구도 필요하다.

이 논문은 한신대학교 학술연구비 지원에 의하여 연구되었음

참고문헌

1. 성정숙·김현철(2015, 1), 국외 컴퓨터 교육과정의 변화 분석, **한국컴퓨터교육학회 논문지, 18**(1), 45-54.
2. 교육부(2015), **초·중등학교 교육과정**[교육부고시 제2015호-80호, 2015.12.1.
3. 성정숙·김현철(2015, 1), 정보교육의 전망과 과제: 미래 정보과 교육과정 개발 방향, **한국컴퓨터교육학회 논문지, 21**(2), 1-10.
4. 김경민(2017, 1), 컴퓨팅 사고력 향상을 위한 정보소양교육에 관한 연구, **한국컴퓨터교육학회 논문지, 20**(4), 59-66.
5. 최현종(2011, 1). 대학 프로그래밍 강좌를 위한 프로그래밍 교육 프레임워크. **한국컴퓨터교육학회 논문지 14**(1), 69-79.
6. 오경선·안성진(2017, 7), 프로그래밍이 어려운 이유와 컴퓨팅 사고력간의 관계성 연구, **컴퓨터교육학회 논문지, 18**(5), 59-62.
7. 오경선·안성진(2016. 3).소프트웨어 교육을 위한 컴퓨팅 사고 교육 내용 설계 기본 연구, **한국컴퓨터교육학회 논문지, 19**(2), 11-20.
8. 최숙영(2015. 5), 초중고에서의 소프트웨어 교육 강화에 따른 문제점과 그 해결방안, **한국컴퓨터교육학회 논문지, 18**(3), 93-104.
9. Kramer, J.(2007, 4), Is abstraction the key to computing? *Communications of the ACM, 50*(4), 37-41.
10. Kernighan, B. & Ritchie, D. (1988), C Programming Language.
11. Dictionary, http://www.dictionary.com/browse/integration.
12. Techtarget, http://searchcrm.techtarget.com/definition/integration.
13. 천인국(2012), 쉽게 풀어쓴 C 언어 Express, 생능출판.
14. 윤성우(2010), 윤성우의 열혈 C 프로그래밍, 오렌지미디어.
15. 김성기(2004, 3), 추상화의 분류, 정보처리학회 논문지, 11-A(1), 89-96.
16. Sommerville, I.(2016), Software Engineering(10 ed.), Pearson.
17. Wikipedia, https://en.wikipedia.org/wiki/Best_coding_practices.
18. Wikipedia, https://en.wikipedia.org/wiki/Subroutine.
19. Wikipedia, https://en.wikipedia.org/wiki/Parameter _(computer_programming).

김성기(Kim, Sung-ki)
1993년: 서울대학교 컴퓨터공학과 졸업
1985년: 동 대학원 컴퓨터공학과 석사
1992년: 동 대학원 컴퓨터공학과 박사
1993년~현재: 한신대학교 컴퓨터공학부 교수
관심분야: 컴퓨터 언어, 데이터베이스, 컴퓨터교육
E-mail: skkim@hs.ac.kr