REPORT

보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.
2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.
3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.
4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과　　목 : 프로그래밍 기초와 실습
과 제 명 : 실습 6: Pointers
담당교수 : 민 형 복 교수
학　　과 : 자연과학계열
학　　년 : 1학년
학　　번 : 2014310407
이　　름 : 이준혁
제 출 일 : 2014년 11월 15일

1, Introduction

이번 실습에서는 포인터에 대해 공부한다. 민형복 교수님께서 프로그래밍 기초와 실습 강의 도중에 "포인터는 여러 가지 이유로 사용되나, 그 중 하나는 함수의 출력을 만들기 위한 것이다." 라는 말씀을 하셨다. 이 말에서 알 수 있듯이, 포인터를 이용하면 함수 내에서 출력을 만들 수 있다. 원래 함수는 return으로 반환하는 출력은 있어도, 인수(argument)에 해당하는 변수의 값 자체를 변경시키는 출력은 없다. 바로 call – by – value를 따르기 때문에 '값'만 넘겨주기 때문이다.

하지만 포인터를 이용하면, 이런 인수에 해당하는 변수의 값 자체를 변경시키는 방법이 있는데, 바로 주소를 이용하는 것이다. 이런 방법으로 함수에서 별도의 출력을 생성할 수 있게 된다. 이것이 바로 포인터를 이용하는 이유 중 하나이다.

이런 기능 때문에 자칫 복잡해질 수 있는 프로그램을 포인터를 이용하면 단순하게 프로그래밍 할 수 있고, 이것이 포인터를 잘 다룰 줄 알아야 하는 이유이다. 그렇기 때문에 이번 실습에서는 4개의 소 문제로 이루어진 포인터 관련 실습을 하면서 포인터가 무엇이며, 어떻게 다루어야 할 지 익히도록 한다.

또 이번 실습의 마지막 소 문제는 static이라는 storage class를 사용해 프로그램을 짜 보는 것이다. static이라는 storage class는 한 파일 내에서 값이 변하지 않는 유형의 변수를 의미한다. 역시 이 static을 이용해 프로그램을 다른

방향으로 짜 볼 수도 있고, 그것이 더 간편할 수도 있기 때문에 숙지하고 있어야 한다.

2. Problem Statement

(A) Describe what the problem is

이번 실습은 네 개의 문제로 구성된다.

문제 1 : Basic Pointer

정수형 변수 num1, num2, num3가 각각 10, 20, 30으로 할당되어 있는 상태에서, 정수형 포인터 타입 pnum1, pnum2, pnum3에 각각 num1, num2, num3의 주소를 할당한다.

이 때 num1, num2, num3의 값을 pnum1, pnum2, pnum3을 이용해 출력하는 것이 이 문제의 목표이다.

문제 2 : swap

두 변수의 값을 바꾸는 swap이라는 함수를 포인터를 이용하지 않고 한번 작성해 보고, 포인터를 이용해서 또 한번 작성해 보는 것이다.

이 때 어떤 차이가 있는 지 확인하고 그 이유를 서술하는 것이 문제에서

요구하는 바 이다.

문제 3 : minimum/maximum

문제 1과 문제 2를 해결하면서 num1, num2, num3에 적당한 값이 부여되었을 텐데, 그 값들 중 최댓값과 최솟값을 구하는 것이 목표이다.

다만, C언어의 함수는 하나의 값만 return할 수 있으므로, 두 개 이상의 출력을 만들기 위해서는 적어도 하나의 변수를 포인터를 이용해서 function의 argument를 변화시키는 식으로 출력해야 한다.

문제 4 : static variable

이전 문제와 비슷하게 num1, num2, num3에 부여된 값들 중 최댓값을 구하는 함수를 작성해야 하는데, static variable이 보존된다는 성질을 이용해야 한다.

또, int maximum_by_static(int number); 의 형태로 생긴 모양의 함수를 세 번 call해야 하는데, 각 function call마다 argument로 num1, num2, num3를 주어야 한다. 이렇게 해서 맨 마지막 return되는 값이 최댓값이 나오도록 함수를 작성해야 한다.

(B) Describe how you solve the problem.

문제 1)

우선 pnum1, pnum2, pnum3의 의미를 이해하는 것이 중요했다. 이들은 모두 정수형 포인터 타입으로, 각각 num1, num2, num3의 주소를 담고 있었다. 그렇기 때문에 pnum1, pnum2, pnum3이 각각 num1, num2, num3을 가리키고 있다고 보았다.

그러면 pnum1, pnum2, pnum3앞에 dereference operator *를 붙이면, 이 값들이 가리키고 있는 값 num1, num2, num3에 접근할 수 있을 것이라 생각했다.

문제 2)

수업시간에 배운 swap 함수를 생각해 보았다. 이 함수는 우선 두 개의 값을 입력 받은 후, 함수 내부에 temp라는 변수를 하나 만들었다. 그리고 두 개의 수를 각각 num1, num2를 이라 두면, 다음과 같이 프로그래밍 하면서 입력 받은 두 개의 변수의 값을 바꾸는 것이었다.

```
temp = num1;
num1 = num2;
num2 = temp;
```

그런데 이 함수는 출력이 없었고, 위처럼 프로그래밍 하면 함수 내부에서 변수들의 값은 바뀔지 몰라도 함수가 끝나면 아무 변화가 없게 될 것이었다고 생각했다. 즉, 포인터를 이용하지 않으면 아무 변화가 없다고 생각해 볼 수 있었다.

하지만 포인터를 이용해, 두 바꾸고자 하는 두 변수의 주소 값을 입력 받아서 포인터로 값을 바꾼다면 이야기가 달라질 것 같았다. 왜냐하면 main함수에 있는 그 변수에 직접 접근해 값을 바꾼다고 보았기 때문이다.

문제 3)

민형복 교수님의 practice6.pdf와 p6.c에 명시된 대로, 최댓값과 최솟값을 구하는 함수에서 최솟값의 주소를 입력 받아서, 최솟값을 포인터를 이용해 출력하기로 했다. 한편, 최댓값은 함수의 반환을 통해 출력하기로 했다.

최댓값을 구하는 과정에서는, 함수 내에서 maximum이라는 변수에 첫 번째 입력 받은 값을 부여하고, if문을 이용해 다른 값이 maximum 보다 크면 그 값을 maximum에 부여하는 식으로 최댓값을 결정하기로 했다. 최솟값의 경우에도 포인터를 이용한다는 점만 제외하면 같은 논리를 이용하면 될 것 같았다.

문제 4)

처음에는 좀 막막해서 민형복 교수님의 p6.c를 참고했다. 그 template에 static 변수 stored_maximum을 -1로 선언하고 시작하는 부분이 나와 있어서, 그것을 이용했다. 문제에서 제시한 함수는 입력 받는 수가 하나였으므로, 입력 받은 수와 stored_maximum을 비교해서, if문을 이용해 더 큰 값을 stored_maximum에 저장한 후에 stored_maximum을 반환하는 형식으로 함수를 만들었다. 그 후 이 함수를 여러 번 call하는 형식으로 세 수의 최댓값을 구할 수 있었다고 보았다.

3. Implementation

main 함수에서 필요한 변수 들은 모두 program template p6.c의 것을 이용하였다.

문제 1)

*pnum1, *pnum2, *pnum3은 각각 pnum1, pnum2, pnum3가 가리키고 있는 주소의 data값, 즉 num1, num2, num3를 의미한다고 볼 수 있다. (pnum1, pnum2, pnum3는 num1, num2, num3의 주소를 가지고 있으므로)

따라서, *pnum1, *pnum2, *pnum3를 출력한다면, num1, num2, num3를 출력한 것이 된다. 이는 문제에서 요구한 것이다.

문제 2)

우선 포인터 없이 만든 함수인 swap함수를 만들어 보았다. 아래는 swap함수에 대한 설명이다.

이 함수는 입력만 있고, 출력은 없다. 다만, 함수 내에서 두 변수의 값을 변경시키고자 하는 것이 출력이 된다. 문제 해결 방법에서 구상한 것과 같이 다음처럼 함수 내부를 작성했다.

```
void swap(int num1, int num2) {
int temp;

temp = num1;
num1 = num2;
num2 = temp;
}
```

마지막 중괄호에서 함수는 끝났으므로, 위 변수들은 모두 삭제되고, 값은 남지 않게 된다. 또, 이 때 main함수의 값이 변경된 것도 없다. 즉, swap함수는 아무런 영향을 끼치지 못한다.

이제 포인터를 이용해 만든 함수인 swap_by_pointer함수를 만들어 보았다.
아래는 swap_by_pointer함수에 대한 설명이다.

이 함수는 swap함수와 기본적인 구성은 비슷하다. 하지만, 입력을 정수 두 개를 받는 것이 아니라 정수형 변수의 주소 두 개를 받아들인다. 또, 함수 내에서는

```
temp = *num1;
*num1 = *num2;
*num2 = temp;
```

와 같이 dereference operator를 이용해 변수 두 개의 값을 바꾼다. 즉, 주소를 준 두 개의 변수가 실제로 값이 달라진다.

즉, swap_by_pointer함수는 출력은 없지만, 포인터를 이용해 main함수의 변수의 값을 변경시키므로, 위 swap 함수와 달리 두 개의 값이 변경된다고 말할 수 있다.

문제 3)

이 문제 역시 최댓값과 최솟값을 결정하는 함수가 필요해서, compute_mixmax라는 이름의 함수를 작성했다. 아래 내용은 이 함수에 대한 설명이다.

구상한 대로 반환 값이 최댓값이고, 포인터를 이용한 출력 값이 최솟값이다. 이를 이용하기 위해서 최솟값을 담는 변수의 주소 값을 입력으로 받아들였다. 물론 비교할 세 개의 수 역시 입력으로 받았다.

그 후 최댓값을 반환할 maximum이라는 정수형 변수를 선언했다. 그 후에 if문을 이용해 값을 비교하면서 최댓값과 최솟값을 가려내었다.

```
*minimum = number1;
if (*minimum > number2)
    *minimum = number2;

if (*minimum > number3)
    *minimum = number3;
```

이렇게 말이다. 그리고 minimum은 이 함수 내에서 포인터 변수이므로 계산하기 위해 dereference operator를 붙여야 했다.

maximum도 dereference operator를 제거하고 부등호만 바꿔서 같은 방법대로 했고, 그 후에 maximum을 반환했다.

이 함수가 반환한 값은 최댓값이므로, main함수에서 function call을 할 때는 다음과 같이 했다.

maximum = compute_mixmax(num1, num2, num3, &minimum);
이렇게 하면 main함수 내의 maximum이라는 변수에 최댓값이 들어가게 되어 올바를 출력을 할 수 있다.

문제 4)

위의 문제들과 마찬가지로 이 함수 역시 함수를 작성하는 것이 중요했다. 그래서 static이라는 변수와 비교해서 최댓값을 구하는 함수인 maximum_by_static이라는 함수를 작성했다. 아래 내용은 이 함수에 대한 설명이다.

이 함수는 여러 변수(여기서는 3개)의 최댓값을 구하지만, 한번에 여러 변수의 입력을 받는 것이 아니라 한번에 입력은 하나씩만 받지만 function call을 여러 번 하면서 최댓값을 구한다는 것에 유의해 입력 받는 값을 하나로 두었다.

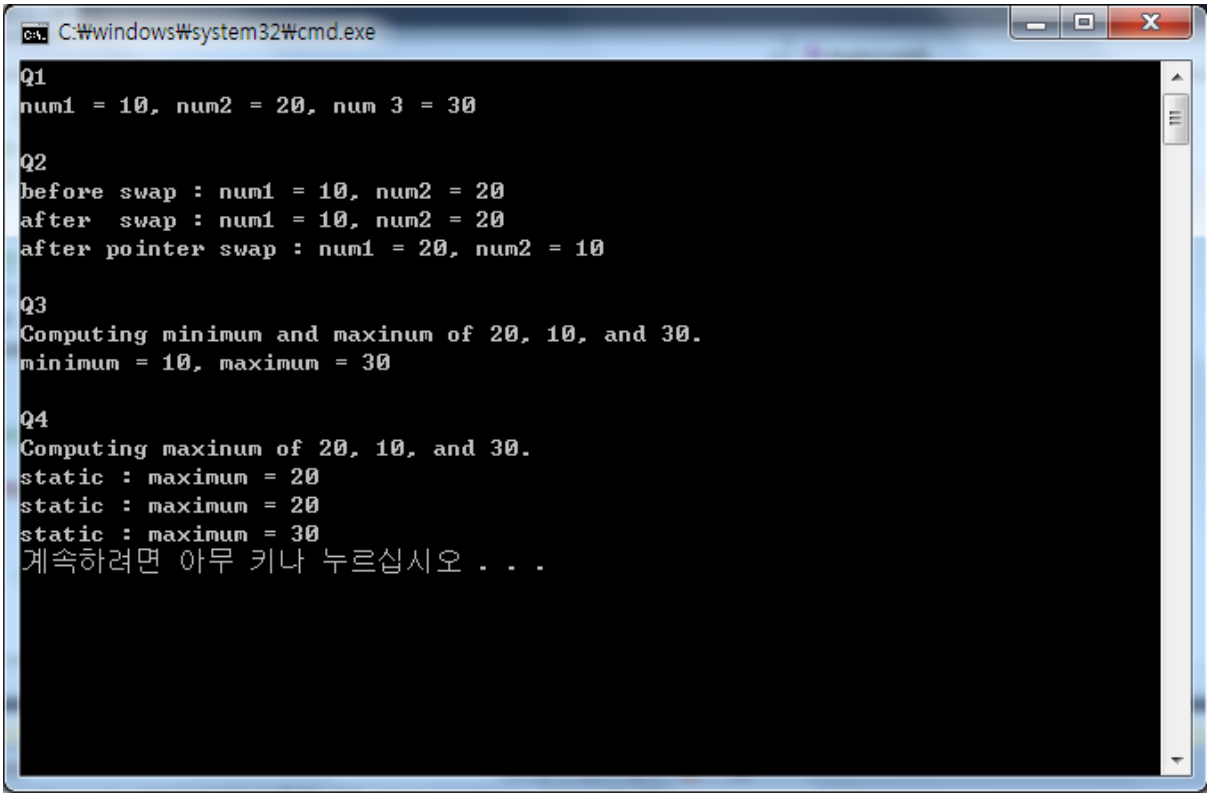
구상한 대로 우선 정수형 static변수 stored_maximum을 -1로 선언하고 시작했다. 그리고 이 stored_maximum과 입력 받은 값을 비교해 입력 받은 값이 더 크다면 stored_maximum에 그 값이 저장되고, 이 값을 반환하는 형식으로 작성했다.

위에서 기술했듯이 이 함수가 여러 번 call되는데, 이 방법을 이용하면
마지막에는 call 될 때 마다 입력 받은 수들 중에서 가장 큰 값을 반환하게 된다.
또한, 각 function call을 할 때마다 반환된 값을 출력함으로 인해 경위를 확인할 수 있게 했다.

물론 이것이 가능했던 이유는 stored_maximum이라는 변수가 static이라
function이 끝나도 값이 없어지거나 변경되지 않았기 때문이다.
또, stored_maximum이 -1이라 입력 받은 세 수 중 어느 수 보다 작기 때문에
절대 최댓값으로 출력되지 않는다는 전제가 있기에 가능했다. 만약
stored_maximum이 40이었다면, 최댓값이 아닌 값인 40이 출력되었을 것이다.

4. Result

이번 실습은 네 문제가 하나로 연결되어 있으므로, 한 화면에 모두 출력했다.



문제 1)

예상한 대로 *pnum1, *pnum2, *pnum3이 num1, num2, num3의 값을 정확하게 출력했다.

문제 2)

그냥 swap function을 이용했을 때에는 변화가 없는 것을 볼 수 있고, 포인터를 이용한 swap을 했을 때는 값이 바뀐 것이 보인다. 예측하는 바와 맞아떨어진다.

문제 3)

num1이 20, num2가 10, num3이 30인데, 최솟값이 10, 최댓값이 30으로 정확하게 나오는 것을 볼 수 있다.

문제 4)

한번 function call을 할 때 마다 반환되는 값을 출력하도록 프로그램을 설계해서 그 과정을 볼 수 있게 하였다.

첫 번째 function call : num 1이 20이므로 20이 출력되어야 정상이다.

두 번째 function call : num 1이 20, num 2가 10이므로 둘 중에 더 큰 20이 출력되어야 정상이다.

세 번째 function call : num 1이 20, num 2가 10, num 3가 30이므로 셋 중에 가장 큰 30이 출력되어야 정상이다.

이렇게 예상한 대로 모두 출력이 되었다.

5. Conclusion & Evaluation

포인터에 대한 실습이 끝났다. 워낙 중요한 단원이라 걱정을 많이 했지만, 실습 문제가 크게 어려운 부분은 없어서 무난히 잘 해결한 것 같다.

포인터라는 것은 정말 새로웠다. 주소를 이용해서 무언가를 하는 것이 굉장히 신선했다. 하지만 처음 배울 때에는 포인터라는 것을 굳이 써야 하나? 라는

의문이 들었다. 그러다 이번 실습의 세 번째 문제를 풀면서 왜 써야 하는 지에 대해 확실하게 감을 잡게 되었다. 한 함수에서 최댓값과 최솟값, 즉 두 개의 출력을 내 놓는 것은 포인터를 쓰지 않으면 불가능하다는 것을 알게 되었기 때문이다. 이 이외에도 이번 실습을 통해 두 변수의 값을 바꿀 때 포인터를 이용하면 굉장히 편하게 함수를 만들 수 있게 된다는 등 포인터의 여러 활용을 몸소 체험할 수 있었다.

그나저나 교수님께서 포인터를 이용하는 다른 이유가 몇 가지 더 있다고 말씀해 주셨고, 또 남은 단원들 대부분에서 포인터를 이용하는 것이 굉장히 많다고 하셨다. 도대체 포인터를 이용해 무엇을 할 수 있을지 상상이 안 되지만, 또 그만큼 무엇을 배울까 기대가 된다. 어쨌든 이번 실습을 통해 포인터를 익히는 데 중요한 초석을 다진 것 같아 굉장히 만족스럽다.

또 storage class중 하나인 static을 이용해 간단한 프로그래밍도 해 보았다. 단지 한 function이 끝나면 사라지는 변수가 아닌 프로그램이 끝날 때까지 변수가 유지된다는 사실은 사뭇 놀라웠고, 그것을 이용해 실제로 프로그래밍도 해 보아서 잘 알게 되었다. 그러나 이 static이라는 유형을 꼭 사용해야 하는 지는 아직 잘 모르겠다. 이것 없이도 충분히 다른 방법으로 프로그램을 짤 수 있을 것 같다. 하지만 extern과 다르게 한 file안에서만 작용하기 때문에 자신이 만든 file 내부에서만 사용한다면 큰 문제는 없으니 이걸 사용하는 것이 편리하다면, 그렇게 하는 것도 괜찮을 것 같다.

6. 참고 문헌

[1] 민형복, program template, p6.c.

[2] 프로그래밍 기초와 실습 사이트, http://class.icc.skku.ac.kr/~min/C/, 보고서 작성 요령, 2014 년 11 월.

[3] Al Kelly, Ira Pohl, *C by Dissection: The Essentials of C Programming, Fourth Edition,* Pearson, p.263 ~ p.280.

[4] 민형복, practice6.pdf.

[5] 민형복, 프로그래밍 기초와 실습 강의