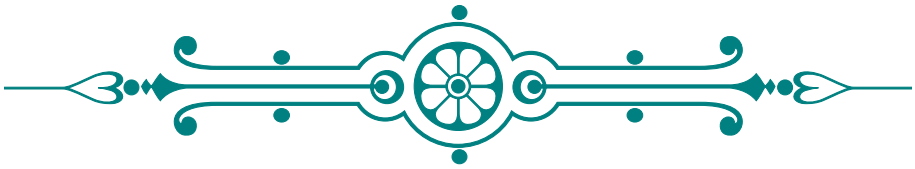

기초 프로그래밍 실습
11월 4주차

학번 : 2016110056

학과 : 불교학부

이름 : 박승원

날짜 : 2016년 11월 25일

7. 직원들의 기본급이 배열 A[]에 저장되어 있다. 배열 B[]에는 직원들의 보너스가 저장되어 있다. 기본급과 보너스를 합하여 이번 달에 지급할 월급의 총액을 계산하고자 한다. A[]와 B[]를 더하여 배열 C[]에 저장하는 함수를 작성하고 테스트하라. 즉, 모든 i에 대하여 C[i] = A[i] + B[i]가 된다.

```
#include<stdio.h>

void array_add(int *A, int *B, int *C, int size) {
    int i;
    for(i=0; i<size; i++) {
        C[i] = A[i] + B[i];
    }
}

int main() {
    int A[10] = {4, 4, 3, 2, 1, 4, 4, 3, 3, 2};
    int B[10] = {4, 5, 3, 7, 1, 6, 2, 3, 3, 2};
    int C[10];

    array_add(A, B, C, 10);
    for(int i=0; i<10; i++) printf("%d ", C[i]);
}
```

```
---------- 문제 7번 실행을 시작합니다. --------------------------
./7.x
8 9 6 9 2 10 6 6 6 4 ---------- 문제 7번 실행을 종료합니다. ----------------------
------
```

8. 직원들의 월급이 배열 A[]에 저장되어 있다고 가정하자. 이번 달에 회사에서 지급할 월급의 총액을 계산하고자 한다. 정수형 배열 원소들의 합을 구하여 반환하는 함수를 작성하고 테스트하라.

```
#include<stdio.h>

int array_sum(int *A, int size) {
    int i, sum=0;
    for(i=0; i<size; i++) {
        sum += A[i];
    }
    return sum;
}

int main() {
    int A[10] = {234, 231, 255, 300, 400, 424, 232, 332, 200, 190};
    printf("  직원들의     월급의     합은  % d입니다 .\n", array_sum(A, 10));
}
```

```
---------- 문제 8번 실행을 시작합니다. --------------------------
./8.x
직원들의 월급의 합은 2798입니다.
---------- 문제 8번 실행을 종료합니다. ----------------------------
```

9. 직원들의 월급이 저장된 배열에서 월급이 200만원인 사람을 찾고 싶을 때가 있다. 주어진 값을 배열 A[]에서 탐색하여 배열 원소의 인덱스를 반환하늟 함수를 작성하고 테스트하라.

```
#include<stdio.h>

int search(int *A, int size, int search_value) {
    int i;
    for(i=0; i<size; i++) {
        if(A[i] == search_value) return i;
    }
}

int main() {
    int A[10] = {234, 231, 255, 300, 400, 424, 232, 332, 200, 190};
    printf("월급이 200 만원인 사람은 % d번째입니다 .\n", 1+search(A, 10, 200));
}
```

```
---------- 문제 9번 실행을 시작합니다. --------------------------
./9.x
월급이 200만원인 사람은 9번째입니다.
---------- 문제 9번 실행을 종료합니다. ----------------------------
```

10.2개의 정수를 입력받아서 최대 공약수와 최소 공배수를 반환하는 함수를 작성하고 테스트하라. 최대 공약수는 유클리드의 방법을 사용하여서 계산한다.

유클리드 호제법(- 互除法, Euclidean algorithm)은 2개의 자연수 또는 정식(整式)의 최대공약수를 구하는 알고리즘의 하나이다. 호제법이란 말은 두 수가 서로(互) 상대방 수를 나누어(除)서 결국 원하는 수를 얻는 알고리즘을 나타낸다. 2개의 자연수(또는 정식) a, b에 대해서 a를 b로 나눈 나머지를 r이라 하면(단, a¿b), a와 b의 최대공약수는 b와 r의 최대공약수와 같다. 이 성질에 따라, b를 r로 나눈 나머지 r'를 구하고, 다시 r을 r'로 나눈 나머지를 구하는 과정을 반복하여 나머지가 0이 되었을 때 나누는 수가 a와 b의 최대공약수이다.

```
#include<stdio.h>

int lcm(int x, int y) {
    if(!x) return y;
    else if(!y) return x;
    int big = x > y ? x : y;
    int small = x < y ? x : y;
    int r = big % small;
    return lcm(small, r);
}

int gcd(int x, int y) {
```

```
    for(int i=1; ; i++) if((x * i) % y == 0) return x * i;
}

void get_lcm_gcd(int x, int y, int *p_lcm, int *p_gcd) {
    *p_lcm = lcm(x, y);
    *p_gcd = gcd(x, y);
}

int main() {
    int x = 36, y = 24, l, g;
    get_lcm_gcd(x, y, &l ,&g);
    printf("%d와 %d의 최대공약수는 %d이고, 최소공배수는 %d입니다.\n", x, y, l, g);
}
```

```
---------- 문제 10번 실행을 시작합니다. --------------------------
./10.x
36와 24의 최대공약수는 12이고, 최소공배수는 72입니다.
---------- 문제 10번 실행을 종료합니다. ----------------------------
```

11. 2개의 정렬된 정수 배열 A[]와 B[]는 똑같은 크기로 정의되어 있다고 가정한다. 배열 C[]에는 충분한 공간이 확보되어 있다고 가정하자. 합치는 알고리즘은 다음과 같다. 먼저 A[0]와 B[0]를 비교한다. 만약 A[0]가 B[0]보다 작으면 A[0]를 C[0]에 복사한다. 다음에는 A[1]과 B[0]를 비교한다. 이번에는 B[0]가 A[1]보다 작다면 B[0]를 C[1]에 저장한다. 똑같은 방식으로 남아있는 원소들을 비교하여 더 작은 원소를 C[]로 복사한다. 만약 A[]나 B[] 중에서 어느 하나가 먼저 끝나게 되면 남아있는 원소들을 전부 C[]로 이동한다. 다음의 그림을 참고하라.

```
#include<stdio.h>

void merge(int* A, int* B, int* C, int size) {
    int a=0, b=0;
    for(int i=0; i<2*size; i++) {
        if(a == size) C[i] = B[b++];
        else if(b == size) C[i] = A[a++];
        else C[i] = A[a] < B[b] ? A[a++] : B[b++];
    }
}

int main()
{
    int A[] = {2,5,7,8};
    int B[] = {1,3,4,6};
    int C[8];

    merge(A, B, C, 4);
    for(int i=0; i<8; i++) printf("%d ", C[i]);
}
```

```
---------- 문제 11번 실행을 시작합니다. --------------------------
./11.x
1 2 3 4 5 6 7 8 ---------- 문제 11번 실행을 종료합니다. --------------------------
--
```