**Introduction to Computer and Lab**

**Homework #6**

**Due date: Mar 19, 2016**

**학번: 201404051**

**이름: 정 용 석**

**1. 조합값 구하기**

**1.1 Solution**

**주어진 3개의 함수, get\_interger(), combination(), factorial() 함수를 이용하여 결과 값을 출력하는 문제이다. 일단 각 함수의 목적과 기능을 살펴보면, 일단 get\_interger() 함수는 간단하게 사용자로부터 입력 받은 수를 반환하는 함수이다. 그래서 반환 형을 보면 int로 말 그대로 사용자에게 입력 받은 int형 자료를 반환 한다. Factorial() 함수 같은 경우는 일단 반환 형은 long long이고 인자는 int n으로서, 이름 그대로 n!(factorial)를 계산하여 반환해 주면 된다. Factorial같은 경우 값이 엄청나게 커지기 때문에 long long 자료형을 이용하여 반환하게 되어있다. 마지막으로 Comination() 함수를 살펴보면, 마찬가지로 반환 형은 long long그리고 인자는 int n과 r을 받는 함수로서, n과 r의 factorial의 조합을 이용하여 계산된 값을 반환해주는 함수이다. 이 조합에 대한 공식은 수업시간에 나온 공식을 그대로 이용하였다. 이 3개의 함수를 사용하여 나온 결과 값을 출력하는 함수로서 computeCombination()함수가 쓰였다. 이는 간단하게 위의 combination함수로 나온 값을 프로그램화 한 함수로서 정리해주는 함수로 보기 쉽다.**

**1.2. Source code**

// 정수값을 입력받고 출력하는 함수

int get\_integer()

{

///////////////////////////////////////

int input;

scanf("%d", &input);

return input;

}

// n! 계산하여 반환하는 함수

long long factorial(int n)

{

///////////////////////////////////////

int i;

long long result = 1;

for (i = 1; i <= n; i++)

result \*= i;

return result;

}

//정수 n과 r의 조합값을 반환하는 함수

long long combination(int n, int r)

{

///////////////////////////////////////

return factorial(n) / (factorial(r)\*factorial(n - r));

}

// 결과 값 출력

void computeCombination()

{

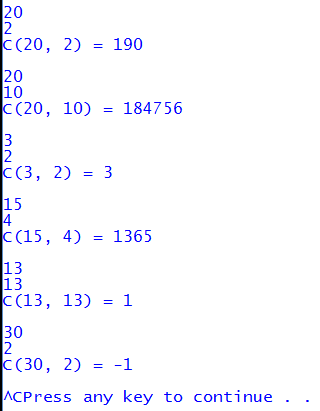
int n = get\_integer();

int r = get\_integer();

printf("C(%d, %d) = %lld\n", n, r, combination(n, r));

}

**1.3. Result (snapshot)**



**2 완전수 구하기**

**2.1. Solution**

**2부터 10000까지의 수 중 완전수를 구하는 문제이다. 간단하게, 숫자를 2부터 10000까지 늘리면서 자기 수를 제외한 약수를 모두 더하여 자기 자신이 되는 수를 찾으면 된다. 이 문제를 위해 2개의 함수가 사용된다. 첫번쨰가 int n이 완전수인지 아닌지를 판별하는 checkPerfect()와 이를 이용하여 2부터 10000까지의 모두 완전수를 구하는 findPerfectNumber()함수이다.**

**2.2. Source code**

// 완전수이면 1, 아니면 0을 반환

int checkPerfect(int n)

{

///////////////////////////////////////

int sum = 0;

int i;

for (i = 1; i< n; i++)

if (n % i == 0)

sum += i;

if (sum == n)

return 1;

else

return 0;

}

// 2~10000 완전수를 출력

void findPerfectNumber()

{

int i;

for (i = 2; i <= 10000; i++)

if (checkPerfect(i))

printf("%d ", i);

printf("\n");

}

**2.3. Result (snapshot)**



**3 10 친화수 구하기**

**3.1. Solution**

**2와 1000 사이의 모든 친화수를 찾는 문제로서 2번과 비슷하게 일단 친화수의 여부를 판단하는 checkFrindNumber함수와 친화수를 찾는 findFrindNumber 함수로 나뉜다. checkFrindNumber함수는 일단 int형 인자를 2개 받는 함수인데, 말 그대로 2개의 정수가 서로 친화수인지만 확인해주면 된다. 따라서 각 정수의 약수를 찾아서 각 약수의 총합이 서로의 값과 동일하면 친화수가 된다. 이를 이용하여 findFrindNumber에서는 2부터 1000까지의 수를 이중 반복 문을 통하여 가능한 경우의 수를 확인, 친화수를 찾게 된다.**

**3.2. Source code**

// 두 수 n1과 n2가 친화수이면 1, 아니면 0

int checkFrindNumber(int n1, int n2)

{

///////////////////////////////////////

int sum1 = 0; //n1의 약수의 총합

int sum2 = 0; //n2의 약수의 총합

int i;

for (i = 1; i < n1; i++)

if (n1 % i == 0)

sum1 += i;

for (i = 1; i < n2; i++)

if (n2 % i == 0)

sum2 += i;

//n1과 n2의 약수의 총합이 서로의 값과 같으면 1

if ((sum1 == n2) && (sum2 == n1))

return 1;

else

return 0;

}

// 2~1000 친화수를 찾는다

void findFrindNumber()

{

int i, j;

for (i = 2; i <= 10000; i++)

{

for (j = i + 1; j <= 10000; j++)

if (checkFrindNumber(i, j))

printf("(%d, %d) ", i, j);

}

printf("\n");

}

**3.3. Result (snapshot)**



**4. 숫자 야구 게임**

**4.1. Solution**

**일반적으로 알고 있는 야구 게임으로서 사용자와 컴퓨터가 서로 대결하는 구도이다. 일단 게임이 시작되면, 프로그램은 자동적으로 3개의 난 수를 만든다. 이 3개의 수는 중복되지 않게 구현되어야 하고, 사용자에 입력에 따라서 스트라이크, 볼, 아웃 여부를 출력해준다. 따라서 스트라이크와 볼을 체크할 수 있는 변수를 만들어주고, 총 9 Round로 진행되므로 이를 체크해 주는 변수 또한 만들어 주었다. 자리를 일치 여부를 판단하기 위해, 컴퓨터가 만든 난 수와 사용자가 입력한 3개의 수 각각의 변수를 따로 선언해주었고, 이를 순차적으로 비교하여 스트라이크와 볼을 if문을 통하여 각각 할당한다. 프로그램이 승패 여부는 9라운드까지 가거나, 혹은 사용자가 3 스트라이크, 즉 모든 수와 위치를 맞추게 되면 끝나기 때문에 while 반복 문을 통하여 구현하였고, 반복 문의 끝에는 항상 변수를 초기화 해주어야 한다. 각각의 상황에 따른 출력 문 또한 다르기 때문에 if문을 통하여 적절하게 편성해주어야 한다. 마지막으로 이 게임은 사용자가 n을 누르기 전까지는 무한으로(사실 상 n이 아닌 다른 것을 입력하여도 프로그램이 종료된다. 이는 사용자가 잘못된 입력 시에 따른 예외 상황을 따로 처리하지 않았다.) 프로그램이 실행이 되기 때문에, 이를 while 무한 반복 문을 통하여 처음에 구현하였는데, 문제가 사용자가 3개의 수를 입력하고 마지막에 누르는 enter키에 있었다. Std input 버퍼에 enter 가 남아있어서, 다음 게임을 위해 사용자가 y나 n을 누르기 전에 이미 scanf를 통해 enter 키가 입력되는 것을 확인할 수 있었다. 따라서 Std input 버퍼에 남아있는 enter 키를 없애기 위해 게임이 종료되고 getchar 함수를 통하여 버퍼에 남아있는 입력을 모두 없애 주고 시작해야 했다.**

**4.2. Source code**

// 숫자 야구게임

void DigitBaseballGame()

{

///////////////////////////////////////

char start; //시작여부

int count = 1; //라운드 횟수

int n1, n2, n3; //컴퓨터 난수 3개

int i1, i2, i3; //사용자 입력 3개

int strike = 0; //스트라이크

int ball = 0; //볼

while (1) {

printf("Play game ? <y/n> :");

scanf("%c", &start);

if (start != 'y') {

printf("Program Closes\n");

break;

}

srand((unsigned)time(NULL));

while (1) //난수 3개 생성, 중복되면 다시

{

n1 = rand() % 10;

n2 = rand() % 10;

n3 = rand() % 10;

if (n1 != n2 && n1 != n3 && n2 != n3)

break;

}

//테스트용 출력

printf("%d %d %d\n", n1, n2, n3);

while (1)

{

printf("\*\*\*\* BASEBALL GAME \*\*\*\* ROUND: %d\n", count++);

printf("USER: ");

scanf("%d %d %d", &i1, &i2, &i3);

//if문을 각각의 일력 수 차례로 비교해서 스트라이크와 볼을 구분해야한다.

//if문 하나에 다 넣게 되면 모든 경우의 수를 뽑아낼 수 없다.

if (n1 == i1)

strike++;

else if (n1 == i2 || n1 == i3)

ball++;

if (n2 == i2)

strike++;

else if (n2 == i1 || n2 == i3)

ball++;

if (n3 == i3)

strike++;

else if (n3 == i1 || n3 == i2)

ball++;

//스트라이크 3개면 사용자의 승리 및 게임 종료

if (strike == 3) {

printf("USER WINNER !!!\n");

printf("The numbers are %d %d %d\n", n1, n2, n3);

//다음 게임을 위한 변수 초기화

strike = 0;

ball = 0;

count = 1;

break;

}

//스트라이크와 볼이 모두 있을 때,

if (strike != 0 && ball != 0)

printf("COMPUTER: %d STRIKE, %d BALL !!!!\n", strike, ball);

//스트라이크만 있을 때

else if (strike != 0 && ball == 0)

printf("COMPUTER: %d STRIKE !!!!\n", strike);

//볼만 있을 때

else if (ball != 0 && strike == 0)

printf("COMPUTER: %d BALL !!!!\n", ball);

//하나도 못 맞추면 OUT

else

printf("COMPUTER: O U T !!!!\n");

//라운드가 끝나면 컴퓨터 승리 및 정답 제공, 프로그램 종료

if (count == 10)

{

printf("\nCOMPUTER WINNER !!!!\n");

printf("The numbers are %d %d %d\n", n1, n2, n3);

//변수 초기화

strike = 0;

ball = 0;

count = 1;

break;

}

printf("\n");

strike = 0;

ball = 0;

}

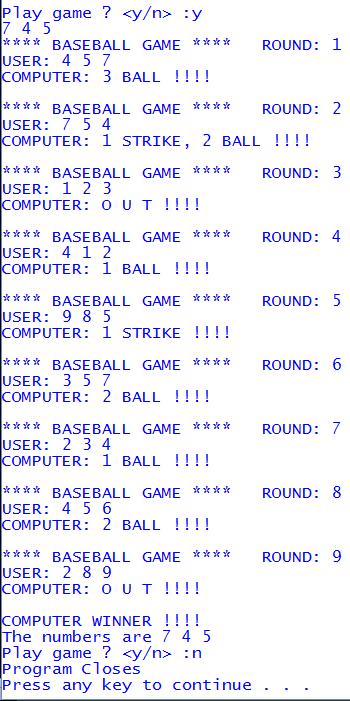
//버퍼에 있는 ENTER 키를 없애기 위한 함수

getchar();

}

}

**4.3. Result (snapshot)**



**5 바이오 리듬 구하기**

**5.1. Solution**

사용자가 생년월일 및 바이오리듬의 년도와 월을 입력하였을 때, 바이오리듬의 결과를 출력하는 프로그램으로, 총 4개의 함수를 구현하여 해결하여야 한다. 첫 번째가 윤년을 판단하는 checkLeapYear()함수로 이는 전 과제에서 많이 구현한 코드를 그대로 사용하였다. 두 번째로 주어진 달이 며칠까지 있는지 판별하는 함수로 이 또한 전 과제 중 달력을 출력하는 함수에서 switch문으로 구현한 코드를 있는 그대로 사용하였다. 생년월일과 입력한 년 월까지의 총 날짜 수를 구하는 countDate함수 같은 경우도 전 과제의 달력 문제와 비슷하기에 구현도 편하다. 한가지 문제가 될 만한 것은 생년월일은 일수까지 입력되기 때문에 그 달의 일 수에서 생년월일의 일 수를 일단 따로 계산해야 된다는 점이다. 그리고 생년월일의 다음 월부터 12월까지의 일 수를 구하고, 그 다음해부터 입력 받은 해까지의 총 일수를 전부 따로 구했다. 물론 다른 방법도 있지만 이렇게 해야 덜 헷갈린 것 같다.

의아했던 점이 printBioRhythm과 computeBioRhythm 함수 였던 것이, 이 2개의 함수의 목적을 바꿔야 하는 것이 아닌가 한다. 하지만 예제에 나와 있듯이 구성과 맞게 구현하기 위해 printBioRhyth 함수에서 각각의 바이오 리듬을 계산하고 출력까지 담당하게 하였고, computeBioRhyth함수에서는 사용자에게 입력을 도맡게 하게 하였다.

**5.2. Source code**

// 주어진 연도가 윤년인지 아닌지 판별하는 함수

int checkLeapYear(int year)

{

///////////////////////////////////////

if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))

return 1;

else

return 0;

}

// 주어진 달이 며칠까지 있는지 판별하는 함수

int checkMonth(int month, int leap\_year)

{

switch (month) {

case 4:

case 6:

case 9:

case 11:

return 30;

break;

case 2://윤년이면 29일, 보통은 28일

if (checkLeapYear(leap\_year))

return 29;

else

return 28;

break;

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

return 31;

break;

}

}

// 생년월일과 년과 월을 입력하였을 때 총 날짜 수 계산

int countDate(int birth\_year, int birth\_month, int birth\_day, int year, int month)

{

///////////////////////////////////////

int i;

int totalDays = 0;

//일단 그 달의 일수 더하기

totalDays += checkMonth(birth\_month, birth\_year) - birth\_day;

//다음달 부터 12월까지 일수 더하기

for (i = birth\_month + 1; i <= 12; i++)

totalDays += checkMonth(i, birth\_year);

//다음해부터 입력된 해 전까지의 일수 더하기

for (i = birth\_year + 1; i < year; i++)

{

if (checkLeapYear(i))

totalDays += 366;

else

totalDays += 365;

}

return totalDays;

}

// 총 날짜 수와 연과 월 입력시 바이오리듬 출력

void printBiorhythm(int date, int year, int month)

{

///////////////////////////////////////

char phys, ment, intel; //신체, 정신, 지능 고조기, 저조기, 위험기 표현 변수

int physP, mentP, intelP; //신체, 정신, 지능 주기

int i;

int days = checkMonth(month, year);

printf("Biorhythm Result:\n");

printf("Low(-), High(\*), Danger(D)\n");

for (i = 1; i <= days; i++)

{

//신체, 정신, 지능 주기 계산

physP = (date+ i - 1) % 23;

mentP = (date + i - 1) % 28;

intelP = (date + i - 1) % 33;

//각각의 저조기, 고조기, 위험기 나누기

if (physP >= 1 && physP <= 10)

phys = '+';

else if (physP >= 13 && physP <= 21)

phys = '-';

else if (physP == 0 || physP == 11 || physP == 12 || physP == 22)

phys = 'D';

if (mentP >= 1 && mentP <= 12)

ment = '+';

else if (mentP >= 15 && mentP <= 26)

ment = '-';

else if (mentP == 0 || mentP == 13 || mentP == 14 || mentP == 27)

ment = 'D';

if (intelP >= 1 && intelP <= 15)

intel = '+';

else if (intelP >= 18 && intelP <= 31)

intel = '-';

else if (intelP == 0 || intelP == 16 || intelP == 17 || intelP == 32)

intel = 'D';

printf("%dM %2dD P: %3d %c\t", month, i, physP, phys);

printf("M: %3d %c\t", mentP, ment);

printf("I: %3d %c\n", intelP, intel);

}

}

// 바이오리듬 계산하기

void computeBiorhythm()

{

int birth\_year, birth\_month, birth\_day;

int year, month, date;

printf("Input a birthdate (ex: 1983 5 21)\n");

scanf("%d %d %d", &birth\_year, &birth\_month, &birth\_day);

printf("Insert wanted year and month. (ex: 2004 11)\n");

scanf("%d %d", &year, &month);

date = countDate(birth\_year, birth\_month, birth\_day, year, month);

printf("\n%d Y %d M %d D and %d Y %d M 1 D has %d days.\n\n", birth\_year, birth\_month, birth\_day, year, month, date + 1);

printBiorhythm(date, year, month);

}

**5.3. Result (snapshot)**

