**C 프로그래밍**

**6 주차 과제**

**제출: 2022년 10월 10일**

**학과: 지능미디어공학과**

**학번: 20221085**

**이름: 김기홍**

가끔은 일정한 시간동안 아무 일도 하지 않으면서 시간을 지연시킬 목적으로 반복문을 사용하기도 한다. 사용자에게서 하나의 수를 입력받아서 변수에 저장한다. 반복문을 사용하여 이 변수의 값을 1씩 감소시키면서 이 변수의 값이 0이 될때까지 반복한다. 반복이 끝나면 벨소리를 낸다.

//딜레이 반복문

#include <stdio.h>

#include <windows.h>

int main(void)

{

int count;

printf("카운터의 초기값을 입력하시오: ");

scanf\_s("%d", &count);

for (count; count > 0; count--)

{

printf("%d ", count);

Sleep(100); //0.1초 지연

}

printf("\a"); //알람발생

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

이 프로그램에서는 사용자가 입력한 수부터 1까지 수가 감소되면서 그 수를 출력한다. 그리고 딜레이를 주기 위해서 windows.h 헤더파일의 Sleep 함수를 사용했다. Sleep 함수의 인수는 ms(milli-second) 단위이다.  
위의 코드에서는 Sleep 함수에 인수로 100을 전달했으므로 반복문이 1회 작동할 때 마다 0.1초 지연된다.  
사용자가 변수에 10을 입력하면 총 1초를 지연한다.

사용자가 입력한 변수가 0이 되면 반복문을 탈출하고 벨소리를 출력한다.  
하지만, "\a"을 사용해도 명령프롬프트에서 소리를 출력하지 않았다. windows.h 헤더에 포함된 Beep 함수를 사용하면 원하는 주파수의 소리를 출력할 수 있었다.

사용자가 입력한 정수의 모든 약수를 화면에 출력하는 프로그램을 작성하라.

//약수 출력

#include <stdio.h>

int main(void)

{

int num;

printf("정수를 입력하시오: ");

scanf\_s("%d", &num);

printf("약수: ");

for (int i = 1; i <= num;i++)

{

if (num % i == 0)

printf("%d ",i);

}

return 0;

}



이 프로그램에서는 사용자가 정수를 입력하면 그 정수의 약수를 모두 출력한다.

약수는 1부터 자기 자신까지가 범위이므로 반복문의 시작값은 1으로, 최대값은 자기 자신(포함)까지 반복 범위를 설정했다. 반복문 안에서는 사용자에게 입력받은 정수의 약수를 구한다. 만약 정수를 1부터 자기 자신까지 나누었을 때 나누어 떨어지면, 정수를 나누었던 수를 출력한다.

60을 입력했을 때, 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60 이 약수로 출력된다.

여기서 곱했을 때 60이 되는 수를 쌍으로 묶으면 다음과 같다.

60이 4로 나누어 떨어진다면, 4는 약수이다.  
60을 4로 나누면 15이고, 이 수도 약수이다.  
그러므로 약수를 1부터 자기자신까지 찾는 것이 아닌, 자기 자신의 제곱근까지만 구하면 나머지 약수도 구할 수 있다.

|  |  |
| --- | --- |
| 1 | 60 |
| 2 | 30 |
| 3 | 20 |
| 4 | 15 |
| 5 | 12 |
| 6 | 10 |

이 방식을 사용해서 프로그램을 최적화해보았다.

약수 출력 최적화 버전

//약수 출력 최적화

#include <stdio.h>

#include <math.h>

int main(void)

{

int num;

int div1[255] = { 0 };

int div2[255] = { 0 };

unsigned char index = 0; //signed를 쓰게 되면 index 역할으로 적절하지 않음.

char exception = 0; //예외 처리용 변수

printf("정수를 입력하시오: ");

scanf\_s("%d", &num);

printf("약수: ");

for (int i = 1; i <= (int)sqrt(num); i++)

{

if (num % i == 0)

{

div1[index] = i;

div2[index] = num / i;

index++;

}

}

index--;

//약수=sqrt(정수)인 경우 예외처리

if (div1[index] == div2[index])

exception = 1;

for (int i = 0; i <= index - exception; i++)//0부터 index 까지

{

printf("%d ", div1[i]);

}

for (int i = index; i >= 0; i--)//index 부터 0까지

{

printf("%d ", div2[i]);

}

return 0;

}

<출력 동일하므로 결과는 생략>

이 프로그램에서 바뀐 점은 약수를 구했을 때, 그 약수의 쌍(서로 곱했을 때 입력받은 정수가 되는 수)도 함께 저장한다. (1의 쌍은 60. 2의 쌍은 30 ···)  
기존에는 입력받은 정수만큼 Mod 연산을 했지만, 이 방식에서는 입력받은 정수의 제곱근(소수자리 버림)만큼만 Mod 연산을 해도 된다.   
단, 나누기 연산이 추가되기는 하지만, 그럼에도 큰 수로 갈수록 이 방식이 더 유리하다.

약수를 하나의 배열에 순서없이 저장하고 마지막에 정렬해도 되지만, 그 경우에는 추가적인 연산시간이 들어간다.

1 60 2 30 3 20 4 15 5 12 6 10 이 경우에는 정렬이 필요함.

(입력받은 수가 60일 때) 약수는 1, 2, 3···6이고 그 약수의 쌍은 60, 30, 20··· 이다 규칙성을 보면 약수가 1, 2, 3으로 커질수록 쌍이 되는 약수의 값은 가장 큰 값에서 점점 작아진다.   
이 규칙성을 이용해서 제곱근 이하의 약수는 1번 배열(1, 2, 3, 4, 5, 6)에 저장하고,  
쌍이 되는 약수는 2번 배열(60, 30, 20, 15, 12, 10)에 저장하고  
2번 배열을 뒤집음(reverse)으로서 복잡한 연산없이 약수를 순서대로 저장할 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1번 배열 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2번 배열 | 60 | 30 | 20 | 15 | 12 | 10 |
| 2번 R | 10 | 12 | 15 | 20 | 30 | 60 |

약수를 저장할 1번 배열과 2번 배열의 길이는 255로 선언했고, 그 길이에 맞는 인덱스도 함께 선언했다.  
인덱스는 unsigned char 자료형을 사용했는데, 이는 0~255의 값을 가지므로 255길이의 인덱스로 적절하다.

그리고 첫번째 프로그램과는 다르게, 두번째 프로그램에서는 중복된 약수를 출력할 가능성이 있다.  
만약 사용자가 입력한 정수가 100이라면 1번 배열에 10이 저장되고, 2번 배열에도 10이 저장되므로 중복된 약수를 출력하게 된다.  
그래서 예외의 경우에 1번 배열에서 값을 하나 덜 출력하도록 함으로써 중복된 약수을 출력하지 않도록 한다.  
그 예외를 관장하는 변수인 ‘expception’을 선언했다. 0혹은 1이면 충분하므로 크기가 작은 char형으로 선언했다.

코드의 최적화를 했으므로 그 성능 차이를 시험해 보았다.   
성능 시험은 사용자에게 값을 받은 직후부터 약수의 출력이 끝난 시점까지.  
ms단위로 측정하였다.  
시간 측정을 위해서 time.h 헤더의 clock\_t 자료형, clock 함수를 사용했다.

1번 프로그램. <https://github.com/Lin-Ion/C/blob/master/6w/6w_2th_code_bench/02.c>

#include <stdio.h>

#include <time.h>

int main(void)

{

clock\_t start, end;

int num;

printf("1번 프로그램\n");

printf("정수를 입력하시오: ");

scanf\_s("%d", &num);

start = clock();

printf("약수: ");

for (int i = 1; i <= num;i++)

{

if (num % i == 0)

printf("%d ",i);

}

end = clock();

printf("\n걸린 시간: %d ms\n",(end - start));

return 0;

}

2번 프로그램. <https://github.com/Lin-Ion/C/blob/master/6w/6w_2th_code_bench/02.1.c>

#include <stdio.h>

#include <math.h>

#include <time.h>

int main(void)

{

clock\_t start, end;

int num;

int div1[255] = { 0 };

int div2[255] = { 0 };

unsigned char index = 0; //signed를 쓰게 되면 index 역할으로 적절하지 않음.

char exception = 0; //예외 처리용 변수

printf("2번 프로그램\n");

printf("정수를 입력하시오: ");

scanf\_s("%d", &num);

start = clock();

printf("약수: ");

for (int i = 1; i <= (int)sqrt(num); i++)

{

if (num % i == 0)

{

div1[index] = i;

div2[index] = num / i;

index++;

}

}

index--;

//약수=sqrt(정수)인 경우 예외처리

if (div1[index] == div2[index])

exception = 1;

for (int i = 0; i <= index - exception; i++)//0부터 index 까지

{

printf("%d ", div1[i]);

}

for (int i = index; i >= 0; i--)//index 부터 0까지

{

printf("%d ", div2[i]);

}

end = clock();

printf("\n걸린 시간: %d ms\n", (end - start));

return 0;

}

시간 출력말곤 변환이 없으므로 6pt로 작게 작성하였습니다.

결과

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

입력: 100,000,000 (1억)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

시간 차이: 최소 340ms

입력: 1,000,000,000 (10억)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

시간 차이: 최소 3350ms

1번 프로그램의 시간 복잡도는 O(n)인 반면, 2번 프로그램의 시간 복잡도는 O()으로 예상됨. (혹은 그 이하)

중첩반복문을 사용하여서 다음과 같이 출력하는 프로그램을 작성하여 보자.

#include <stdio.h>

int main(void)

{

int n;

printf("정수를 입력하시오: ");

scanf\_s("%d", &n);

for (int i = 1; i <= n; i++) //1부터 n 까지 반복

{

for (int j = 1; j <= i; j++) //1부터 i 까지 반복

{

printf("%d ", j);

}

printf("\n"); //한줄의 출력 끝

}

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

이 프로그램은 사용자가 입력한 수만큼 1에서 하나씩 증가하는 값을 붙여가며 출력한다.  
이중 반복문을 사용했다. 1차 반복문에서는 개행을 담당하고, 2차 반복문에서는 한 줄의 출력을 담당한다.

2와 100사이에 있는 모든 소수(prime number)를 찾는 프로그램을 작성하라. 정수가 소수가 되려면 1과 자기 자신만을 약수로 가져야 한다.

#include <stdio.h>

#define n 101

int main(void)

{

int prime[n] = {0}; //0부터 n까지의 배열을 모두 0으로 초기화함

for (int i = 2; i\*i < n; i++)

{

if (prime[i] == 1)

continue;

for (int j = i\*2; j < n; j += i)

prime[j] = 1;

}

for (int i = 2; i < n; i++)

{

if (prime[i] == 0)

printf("%d ", i);

}

return 0;

}

이 프로그램은 1부터 100까지의 수 중에서 소수를 출력한다.

소수인지를 판별하기 위해 1부터 자기 자신까지 나누어서 1와 자기자신이외의 약수를 가지지 않으면, 즉 나누어 떨어지지 않으면 소수이다. 가장 간단한 코드를 작성한다면 1부터 100까지 반복하는 반복문 내부에 그 수가 소수인지를 판단하는 반복문을 작성한다. 이러한 코드는 하나의 수가 소수인지를 판별하는 데에는 유리하지만,  
소수를 나열하는 경우에는 에라토스테네스의 체를 사용한 알고리즘이 유리하다.

어떤 수가 2이상의 수의 배수라는 것은 소수가 아니라는 말이므로, 소수를 찾고자 하는 범위의 제곱근 이하에서 2부터 수를 1씩 늘려가며 그 수의 배수를 모두 찾는다.

(1+2+3+…+n)가 10000을 넘지 않으면서 가장 큰 값과 그 때의 n의 값을 구하라.

#include <stdio.h>

int main(void)

{

int sum = 0;

int n = 1;

while (1) //무한반복

{

sum += n;

if (sum > 10000)

{

sum -= n;

n--;

break;

}

n++;

}

printf("1부터 %d까지의 합이 %d입니다.\n", n, sum);

return 0;

}

무한 반복을 위해서 while 문의 조건이 항상 참이 되도록 상수 값을 넣었고  
 n이 1부터 반복마다 1씩 증가하며 sum 변수에 n을 더한다.  
만약 sum이 10000을 넘는다면 반복문을 종료하고 넘기 직전의 sum값과 n 값으로 돌아간다.

**과제 작성 중, 과제 범위가 정정(258p -> 315p)되었기 때문에… 이미 작성했던 코드들만 올려둡니다.**

**아래는 정정 이전의 과제범위입니다.**

키보드에서 하나의 문자를 읽어서 모음과 자음을 구분하는 프로그램을 작성하여 보자. 단 switch 문을 사용한다.

//모음 자음 구분

#include <stdio.h>

int main(void)

{

char eng;

printf("문자를 입력하시오: ");

eng = getchar();

switch (eng)

{

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

case 'y':

printf("모음입니다.\n");

break;

default:

printf("자음입니다.\n");

break;

}

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자로부터 3개의 정수를 읽어 들인 후에 if-else 문을 사용하여 가장 작은 값을 결정하는 프로그램을 작성하라.

//가장 작은 값

#include <stdio.h>

int main(void)

{

int x,y,z; // x,y,z

int min = 0;

printf("3개의 정수를 입력하시오: ");

scanf\_s("%d %d %d", &x, &y, &z);

if (x < y)

min = x; // x<y 라면 x 저장

else

min = y; // y<x 라면 y 저장

if (z < min)

min = z; //x, y중 낮은 값을 z와 비교

printf("제일 작은 정수는 %d입니다.\n",min);

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

//가장 작은 값 list

#include <stdio.h>

int main(void)

{

int ary[255] = {0};

char min = 0;

int n = 0;

printf("정수의 개수를 입력하세요: ");

scanf\_s("%d", &n);

for (int i = 0; i < n; i++)

{

printf("정수를 입력하세요(%d/%d): ", i+1, n);

scanf\_s("%d", &ary[i]);

}

for (int i = 1; i < n ; i++)

{

if (ary[i] < ary[min])

min = i;

}

printf("제일 작은 정수는 %d입니다.\n", ary[min]);

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

놀이 공원에서 롤러코스터에 타려면 키가 140츠 이상이고 나이가 10살 이상이어야 한다고 가정하자. 사용자에게 키와 나이를 질문한 후에 “타도 좋습니다” 또는 “죄송합니다”를 출력하는 프로그램을 작성하여 보자.

//놀이공원 탑승조건

#include <stdio.h>

int main(void)

{

int height;

int age;

printf("키를 입력하시오(cm): ");

scanf\_s("%d", &height);

printf("나이를 입력하시오: ");

scanf\_s("%d", &age);

if (height >= 140 && age >= 10) // 키 140이상 나이 10세 이상

printf("타도 좋습니다.\n");

else

printf("죄송합니다.\n");

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자로부터 키를 입력받아서 표준 체중을 계산한 후에 사용자의 체중과 비교하여 저체중인지, 표준인지, 과체중인지를 판단하는 프로그램을 작성하라. 표준 체중 계산식은 다음을 사용하라.

//표준 체중

#include <stdio.h>

int main(void)

{

float weight;

float height;

float std\_weight; //표준체중

float std\_per\_weight; //표준체중 대비 체중의 백분율

printf("키와 체중을 입력하세요: ");

scanf\_s("%f %f", &height,&weight);

std\_weight = (height - 100) \* 0.9; // (키-100) x 0.9

std\_per\_weight = (weight / std\_weight) \* 100; // (체중/표준체중)\*100

if (std\_per\_weight < 90) //표준 대비 90% 미만

printf("저체중입니다.\n"); //저체중

else if (std\_per\_weight <= 110) //표준 대비 90%이상 110% 이하

printf("표준체중입니다.\n");//표준체중

else //표준 대비 110% 초과

printf("과체중입니다.\n"); //과체중

printf("표준 체중 대비 백분율: %f\n", std\_per\_weight);

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같이 정의되는 함수의 함수값을 계산하여 보자. 사용자로부터 x값을 입력받아서 함수값을 계산하여 화면에 출력한다. x는 실수이다.

//다항 함수 계산

#include <stdio.h>

#include <math.h>

int main()

{

float x, y;

printf("{ x²-9x+2 (x≤0)\n{ 7x+2 (x<0)\n");

printf("x의 값을 입력하시오: ");

scanf\_s("%f", &x);

if (x <= 0)

y = pow(x, 2) + (-9 \* x) + (2); //x^2 -9x +2

else

y = (7 \* x) + (2); //7x +2

printf("f(x)의 값은 %f\n",y);

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

사용자로부터 하나의 문자를 입력받아서 문자가 ‘R’이나 ‘r’이면 “Rectangle”이라고 출력한다. “T이거나 ‘t’이면 “Triangle”, ‘C’이거나 ‘c’이면 ‘Circle”이라고 출력하는 프로그램을 작성한다. 그 외의 문자가 들어오면 “Unknown”이라고 출력한다.

//도형 출력//

#include <stdio.h>

int main(void)

{

char shape;

printf("문자를 입력하시오: ");

shape = getchar() & ~0x20;

//32의 1의 보수(1011111)을 사용해서 소문자를 대문자로 변환

//32비트 자리의 1을 0으로 만듬.

switch (shape)

{

case 'R':

printf("Rectangle");

break;

case 'T':

printf("Triangle");

break;

case 'C':

printf("Circle");

break;

default:

printf("Unknown");

break;

}

return 0;

}

텍스트이(가) 표시된 사진

자동 생성된 설명

/\*

 bin(32)

'0b0100000'

bin(ord('A'))

'0b1000001'

대문자를 소문자로 바꾸고 싶다면,

32에 해당하는 비트를 1으로.

1000001

OR 0100000

  = 1100001  = 'A'

bin(ord('a'))

'0b1000001'

소문자에 같은 연산을 한 경우.

1100001

OR 0100000

  = 1100001  = 'a'

/\*

 bin(32)

'0b0100000'

bin(ord('a'))

'0b1100001'

소문자를 대문자로 바꾸고 싶다면,

32에 해당하는 비트를 0으로.

1100001

AND 1011111

  = 1000001  = 'A'

bin(ord('A'))

'0b1000001'

대문자에 같은 연산을 한 경우.

1000001

AND 1011111

  = 1000001  = 'A'

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

 /\*

마스킹 결과로 대문자 'A' 또는 소문자 'a' 가 나오는 경우는

 입력값이 'A' 또는 'a' 인 경우 밖에 없음을 파이썬에서 귀납적으로 실험함.

\*/

mask=0x20

print("대문자 A가 나오는 경우")

for i in range(1,100000):

    if ~mask&i == ord('A'):

        print(f'i={i} chr({i})={chr(i)}')

print("\n소문자 a가 나오는 경우")

for j in range(1,100000):

    if mask|j == ord('a'):

        print(f'j={j} chr({j})={chr(j)}')

print(f'\ni={i} j={j}')

텍스트이(가) 표시된 사진

자동 생성된 설명