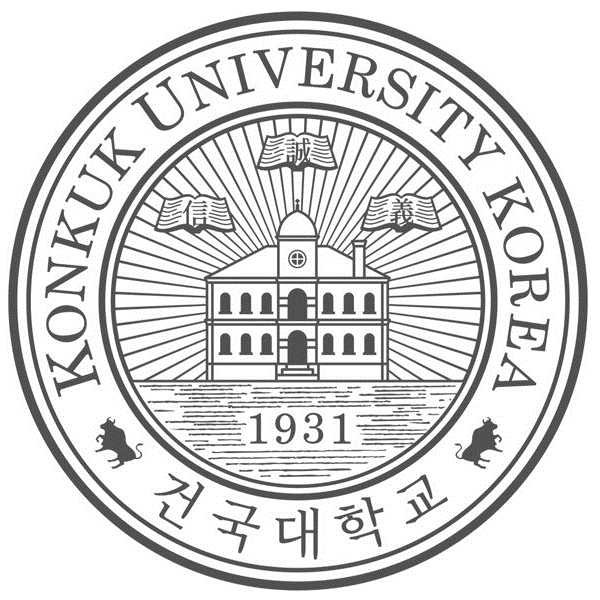
****

2020년 4월 29일

이호은

202011353

컴퓨터공학부

C프로그래밍

0451분반

과 제

Programming Assignment 1

분반

담당교수님

학과

학번

제출일

이름

과목명

박소영 교수님

**Programming Assignment 1**

**Question 1. 사각 테이블 배치 문제**

1) 문제 정의

방의 가로, 세로 크기, 테이블의 가로, 세로 크기, 테이블 간 간격, 테이블 당 하객 수를 입력 받고 방과의 테이블 간격, 테이블 사이의 간격을 만족했을 때의 최대 수용 인원을 계산하여 출력한다.

|  |  |
| --- | --- |
| 입력 조건 | 출력 조건 |
| - 방의 가로 및 세로 크기는 미터 단위로 양의 정수이며 1000보다는 작다고 가정한다.  - 테이블의 가로 및 세로 크기는 미터 단위로 양의 정수이며 1000보다는 작다고 가정한다.  - 테이블 간 간격은 미터 단위로 양의 정수이며 10보다 작다고 가정한다.  - 테이블당 하객 수는 양의 정수로 3명이상 20명 이하라고 가정한다.  - 각 입력 값의 범위는 프로그램에서 검사할 필요는 없으며, 사용자가 값을 입력할 때, 주어진 조건에 맞는 값을 입력한다고 가정한다. 단, 변수 선언을 위한 자료형은 입력 값의 범위를 고려하여, 최소의 메모리를 사용할 수 있도록 선언한다. | 프로그램은 주어진 조건에 따른 최대 하객 수를 출력한다. 결과는 반드시 다음의 출력 형태를 따르며, X로 표시된 부분은 프로그램 실행에 의해 계산된 값이다.  This arrangement seats X people. |

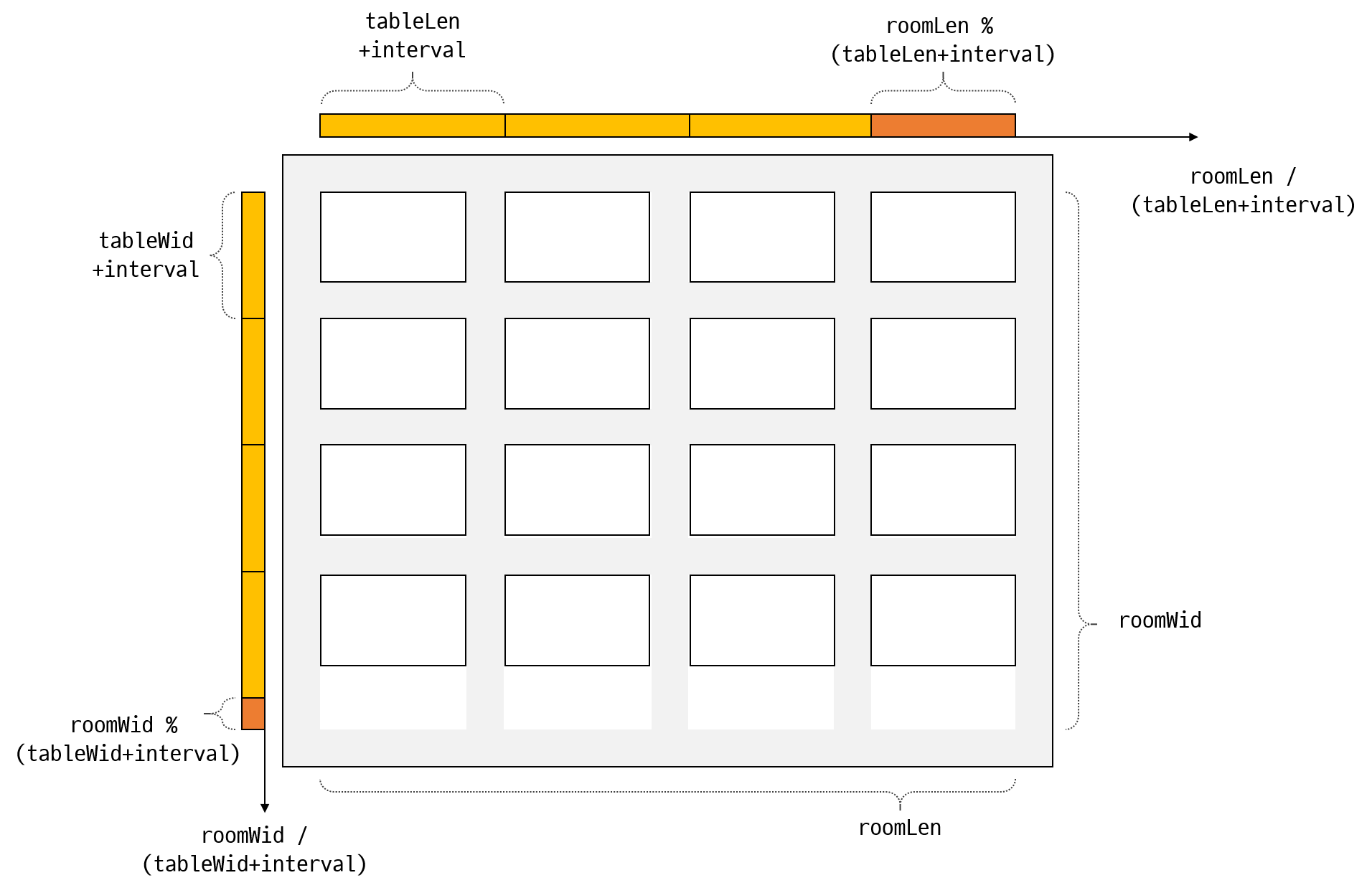
2) 주요 변수 설명

|  |  |  |
| --- | --- | --- |
| 변수명 | 자료형 | 설명 |
| **roomLen** | int | 방의 가로 길이(length)를 저장한다. (m) |
| **roomWid** | int | 방의 세로 길이(width)를 저장한다. (m) |
| **tableLen** | int | 테이블 한 개의 가로 길이(length)를 저장한다. (m) |
| **tableWid** | int | 테이블 한 개의 세로 길이(width)를 저장한다. (m) |
| **interval** | unsigned char | 테이블과 테이블 사이, 방 끝과 테이블 사이의 간격을 저장한다. (m) |
| **person** | int | 테이블 당 사람 수, 나중에는 이 변수값과 테이블 수를 곱하여 전체 사람 수를 저장한다. |
| **temp** | int | 변수 interval의 값을 입력 받을 때 사용.\* |

- unsigned char의 범위는 0~255까지의 수를 저장할 수 있으며 정수를 저장하는 변수 중에선 가장 크기가 작다. 즉, 범위가 1~10인 interval과 3~20인 person의 값을 저장하기에 용이하다. 하지만 person 변수는 나중에 최종적인 하객 수를 저장하게 되므로 255를 넘어갈 수 있으므로 정수형으로 선언해주었다.

\* scanf(“%d”, &interval); 과 같이 unsigned char 변수에 scanf 함수로 10진수 값을 입력 받아 저장하자 프로그램 실행 도중 에러가 발생하였습니다. 이를 해결하기 위해 정수형 변수에 값을 입력 받은 후 이 값을 다시 각 변수에 대입해주는 방식으로 해결해주었다.

3) 아이디어 및 알고리즘



- 총 테이블의 수는 가로로 배치되는 최대 테이블의 수와 세로로 배치되는 최대 테이블 수를 곱한 값이다. 총 하객 수는 총 테이블 수에 테이블 당 사람 수를 곱한 값이다.

- 먼저 방의 사면의 벽에는 테이블과 간격이 있어야 한다. 실질적으로 테이블이 놓일 수 있는 방의 크기는 가로 roomLen – 2 \* interval, 세로는 roomWid – 2 \* interval이다.

- 실질적 방의 테이블이 가로, 세로로 각각 몇 개가 들어가는지 확인하기 위해서는 테이블의 가로, 세로 길이와 간격을 더한 값을 하나의 테이블로 생각하면 간격이 있을 때의 테이블의 최대 개수를 구할 수 있다.

- roomWid / (tableWid + interval)과 roomLen / (tableLen + interval)로 가로, 세로로 들어가는 테이블의 수를 구할 수 있다.(여기서 roomWid와 roomWid는 실질적 방의 가로, 세로 길이이다. 즉 roomLen = roomLen – 2interval, roomWid = roomWid – 2interval이다.)

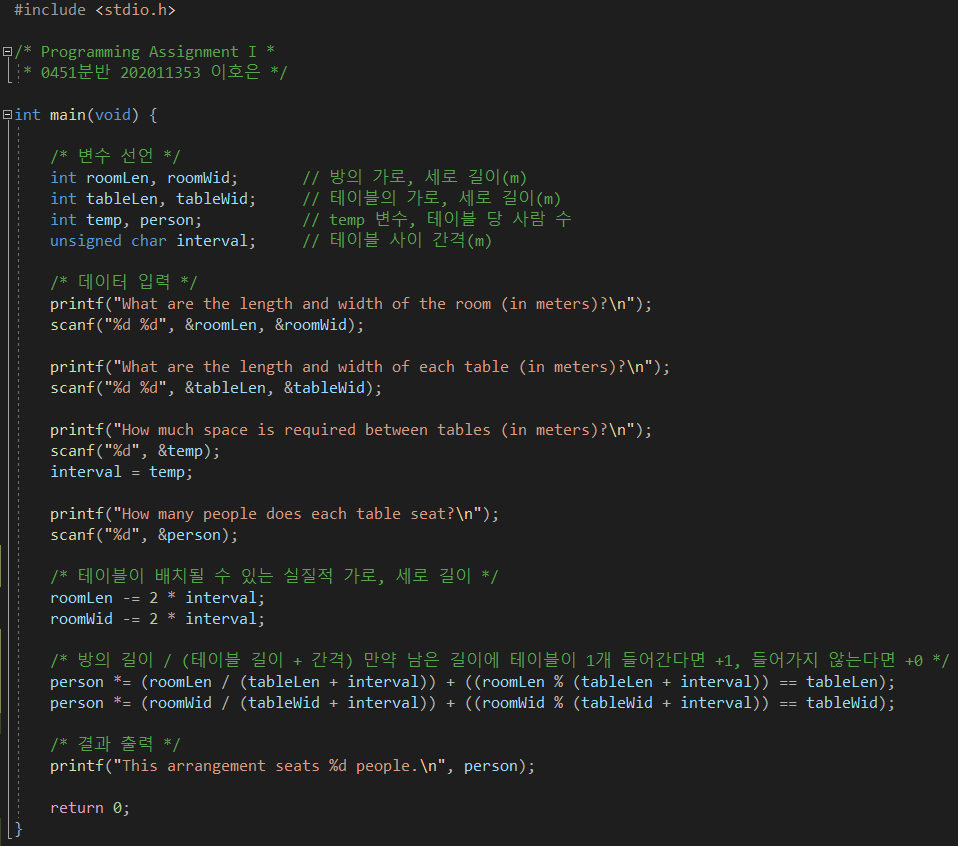
- 여기서 문제가 각 마지막 줄에 배치되는 테이블의 경우 간격을 더하게 되면 끝쪽에는 방으로부터의 간격과 테이블로부터의 간격, 즉 간격이 2개가 생기게 된다. 즉 남은 공간 (방의 길이 / (테이블 길이 + 간격))의 나머지 즉, (방의 길이 % (테이블의 길이 + 간격)) 내에 테이블이 들어갈 수 있는지를 확인해야 한다.

- 이때, 테이블의 길이보다 나머지가 크다면 몫이 +1이 될 것이므로 나머지의 최댓값은 테이블의 길이와 같다. 즉, 나머지 공간에 테이블이 한 줄 더 들어갈 수 있는 경우는 (나머지 = (방의 길이 % (테이블의 길이 + 간격)) == 테이블의 길이)가 될 것이다. 이때, 만약 나머지 공간에 테이블이 들어갈 수 있다면 조건식이 참이 되어 +1이 될 것이고 테이블이 들어갈 수 없다면 조건식이 거짓이 되어 +0이 될 것이다.

- 알고리즘

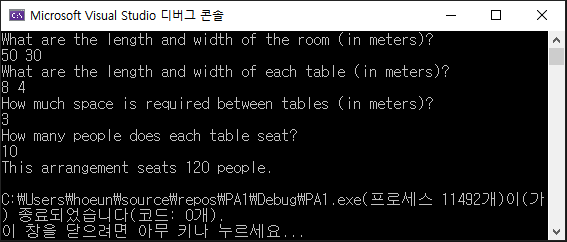
|  |
| --- |
| ⓐ 데이터를 입력 받는다.  ⓑ 실제로 테이블이 배치될 수 있는 방의 가로, 세로 길이를 각각 구한다. = (방 길이 – 2 \* 간격)  ⓒ 실질적 방 가로 길이 / (테이블 가로 길이 + 간격)  ⓓ 만약 ⓒ에서 테이블이 배치되고 남은 길이에 테이블이 들어갈 수 있다면 +1  ⓔ 이를 person 변수의 값과 곱하여 person 변수에 대입한다. (사람수 x 가로 개수)  ⓕ 실질적 방 세로 길이 / (테이블 세로 길이 + 간격)  ⓖ 만약 ⓕ에서 테이블이 배치되고 남은 길이에 테이블이 들어갈 수 있다면 +1  ⓗ 이를 person 변수의 값과 곱하여 person 변수에 대입한다. (사람수 x 가로 개수 x 세로 개수)  ⓘ 결과를 출력한다. |

4) 소스코드



5) 수행 결과

- 주어진 Sample Data를 이용한 수행



- 데이터 셋을 통한 프로그램 검증

|  |  |  |
| --- | --- | --- |
| 1 | 데이터 설정 | |
| int roomLen = 1000, roomWid = 1000  int tableLen = 1, tableWid = 1  int interval = 1, person = 20 | |
| 프로그램 출력 결과 | |
|  | 4,980,020명 |
| 검증 | |
| (최대의 경우)  1000m x 1000m의 방 안에 1m x 1m의 테이블이 1m 간격으로 배치되면 이와 같은 모습이다.  가로로 테이블은 간격, 테이블, 간격, 테이블, …, 테이블, 간격 순으로 놓이게 된다. 1000m일 때 1m의 간격을 두고 1m 길이의 테이블은 499개가 들어갈 수 있다. (998 / 2)  가로 499개 x 세로 499개 = 249,001개이다.  249,001개의 테이블에 20명씩 앉는다면 249,001개 x 20명 = 4,980,020명이다. | |
| 2 | 데이터 설정 | |
| int roomLen = 100, roomWid = 100  int tableLen = 80, tableWid = 80  int interval = 10, person = 1 | |
| 프로그램 출력 결과 | |
|  | 1명 |
| 검증 | |
| (정확히 일치하는 경우)  100m x 100m의 방 안에 80m x 80m 테이블이 10m 간격으로 배치되면 이와 같은 모습이다.  실제로 테이블이 배치될 수 있는 공간의 크기는 가로, 세로 길이가 100m – 2 \* 10m = 80m이다.  80m x 80m의 공간 안에 80m x 80m 크기의 1개 테이블이 들어갈 수 있다. (정확히 일치)  1개의 테이블에 1명이 앉는다면 1명이다. | |
| 3 | 데이터 설정 | |
| int roomLen = 10, roomWid = 10  int tableLen = 100, tableWid = 100  int interval = 10, person = 10 | |
| 프로그램 출력 결과 | |
|  | 0명 |
| 검증 | |
| (초과의 경우)  10m x 10m 방에 100m x 100m 크기의 테이블이 들어갈 수 없으므로 0명이다. | |
| 4 | 데이터 설정 | |
| int roomLen = 100, roomWid = 100  int tableLen = 50, tableWid = 50  int interval = 10, person = 1 | |
| 프로그램 출력 결과 | |
|  | 1명 |
| 검증 | |
| 100m x 100m의 방 안에 50m x 50m 테이블이 10m 간격으로 배치되면 이와 같은 모습이다.  실제로 테이블이 배치될 수 있는 공간의 크기는 가로, 세로 길이가 100m – 2 \* 10m = 80m이다.  80m x 80m의 공간 안에 50m x 50m 크기의 1개 테이블이 들어갈 수 있다. (남은 공간의 가로, 세로 길이는 30m 이므로 테이블이 들어갈 수 없다.)  1개의 테이블에 1명이 앉는다면 1명이다. | |
| 5 | 데이터 설정 | |
| int roomLen = 200, roomWid = 300  int tableLen = 5, tableWid = 10  int interval = 3, person = 1 | |
| 프로그램 출력 결과 | |
|  | 528명 |
| 검증 | |
| 가로의 경우 194m(200m – 2\*3m) 안에 5m 길이의 테이블이 3m의 간격을 두고 24개가 들어갈 수 있다. (194 / (5 + 3)) 나머지는 2m이므로 5m의 테이블이 들어갈 수 없다.  세로의 경우 294m(300m – 2\*3m) 안에 10m 길이의 테이블이 3m의 간격을 두고 22개가 들어갈 수 있다. (294 / (10 + 3)) 나머지는 8m이므로 10m의 테이블이 들어갈 수 없다.  가로 24개 x 세로 22개 = 528개이다. | |

6) 토의 사항

- 그동안 프로그래밍을 하면서 연산식 내에 조건식(부등식, 등식 등)을 이용해서 연산을 한 적은 없었는데 이번 프로그래밍을 하며 연산식 내에 조건식을 도입한다면 더욱 연산이 간단해진다는 것을 배웠습니다.

- 사실 이번 코딩을 하며 거의 1주일 가까이 시간이 걸렸습니다. 그동안 프로그래밍을 하며 수학적인 연산에 관련된 프로그래밍을 많이 접해보았지만 거의 수학 자체에 가까운 프로그램이다보니 알고리즘을 여러 번 수정하며 프로그램을 짰습니다. 많은 시도 끝에 돌고 돌아 코드가 처음과 거의 유사한 형태가 되어서 약간 놀라기도 했습니다. 프로그래밍을 할 때, 문법 요소를 잘 아는 것도 중요하지만 얼마나 수학적 알고리즘을 효율적으로 빠르게 짤 수 있느냐가 중요하다는 것을 배웠습니다.

**Question 2. 원탁 테이블 배치 문제**

1) 문제 정의

방의 가로, 세로 크기, 테이블의 반지름 길이, 테이블 간 간격, 테이블 당 하객 수를 입력 받고 방과의 테이블 간격, 테이블 사이의 간격을 만족했을 때의 최대 수용 인원을 계산하여 출력한다.

|  |  |
| --- | --- |
| 입력 조건 | 출력 조건 |
| - 방의 가로 및 세로 크기는 미터 단위로 양의 정수이며 1000보다는 작다고 가정한다.  - 테이블의 반지름의 길이는 미터 단위로 양의 정수이며, 1에서 500사이의 값이다.  - 테이블 간 간격은 미터 단위로 양의 정수이며 10보다 작다고 가정한다.  - 테이블당 하객 수는 양의 정수로 3명이상 20명 이하라고 가정한다.  - 각 입력 값의 범위는 프로그램에서 검사할 필요는 없으며, 사용자가 값을 입력할 때, 주어진 조건에 맞는 값을 입력한다고 가정한다. 단, 변수 선언을 위한 자료형은 입력 값의 범위를 고려하여, 최소의 메모리를 사용할 수 있도록 선언한다. | 프로그램은 주어진 조건에 따른 최대 하객 수를 출력한다. 결과는 반드시 다음의 출력 형태를 따르며, X로 표시된 부분은 프로그램 실행에 의해 계산된 값이다.  This arrangement seats X people. |

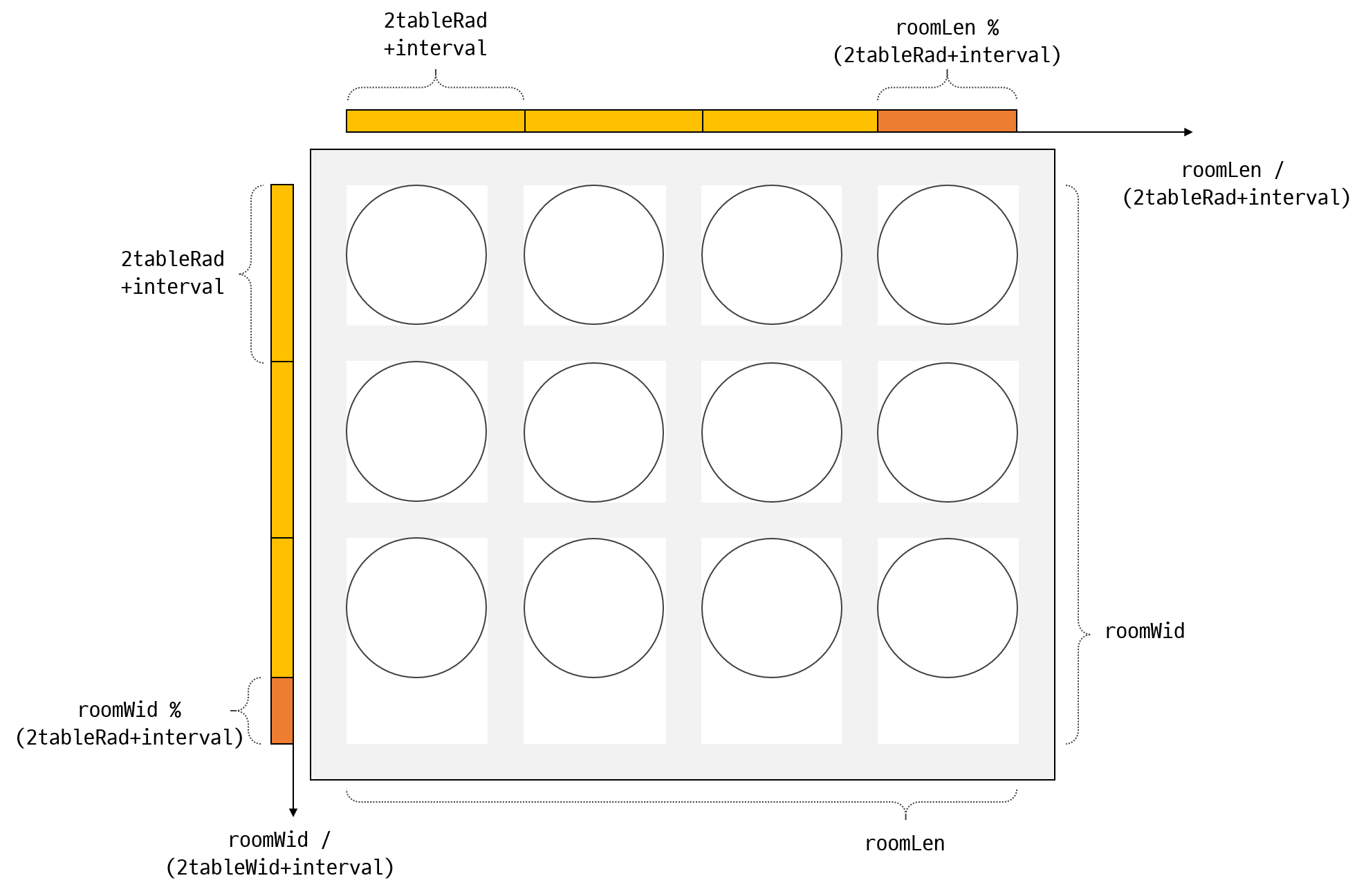
2) 주요 변수 설명

|  |  |  |
| --- | --- | --- |
| 변수명 | 자료형 | 설명 |
| **roomLen** | int | 방의 가로 길이(length)를 저장한다. (m) |
| **roomWid** | int | 방의 세로 길이(width)를 저장한다. (m) |
| **tableRad** | int | 테이블 한 개의 반지름 길이를 저장한다. (m) |
| **interval** | unsigned char | 테이블과 테이블 사이, 방 끝과 테이블 사이의 간격을 저장한다. (m) |
| **person** | int | 테이블 당 사람 수, 나중에는 이 변수값과 테이블 수를 곱하여 전체 사람 수를 저장한다. |
| **temp** | int | 변수 interval의 값을 입력 받을 때 사용.\* |

- unsigned char의 범위는 0~255까지의 수를 저장할 수 있으며 정수를 저장하는 변수 중에선 가장 크기가 작다. 즉, 범위가 1~10인 interval과 3~20인 person의 값을 저장하기에 용이하다. 하지만 person 변수는 나중에 최종적인 하객 수를 저장하게 되므로 255를 넘어갈 수 있으므로 정수형으로 선언해주었다.

\* scanf(“%d”, &interval); 과 같이 unsigned char 변수에 scanf 함수로 10진수 값을 입력 받아 저장하자 프로그램 실행 도중 에러가 발생하였습니다. 이를 해결하기 위해 정수형 변수에 값을 입력 받은 후 이 값을 다시 각 변수에 대입해주는 방식으로 해결해주었다.

3) 아이디어 및 알고리즘

****

- 총 테이블의 수는 가로로 배치되는 최대 테이블의 수와 세로로 배치되는 최대 테이블 수를 곱한 값이다. 총 하객 수는 총 테이블 수에 테이블 당 사람 수를 곱한 값이다.

- 먼저 방의 사면의 벽에는 테이블과 간격이 있어야 한다. 실질적으로 테이블이 놓일 수 있는 방의 크기는 가로 roomLen – 2 \* interval, 세로는 roomWid – 2 \* interval이다.

- 1번 문제인 사각 테이블 배치 문제와 알고리즘은 일치한다. 다만 테이블의 가로, 세로 길이가 아닌 테이블의 지름으로 계산해야 한다.

- 실질적 방의 테이블이 가로, 세로로 각각 몇 개가 들어가는지 확인하기 위해서는 테이블의 지름 길이와 간격을 더한 값을 하나의 테이블로 생각하면 간격이 있을 때의 테이블의 최대 개수를 구할 수 있다.

- roomWid / (tableRad + interval)과 roomLen / (tableRad + interval)로 가로, 세로로 들어가는 테이블의 수를 구할 수 있다.(여기서 roomWid와 roomWid는 실질적 방의 가로, 세로 길이이다. 즉 roomLen = roomLen – 2interval, roomWid = roomWid – 2interval이다.)

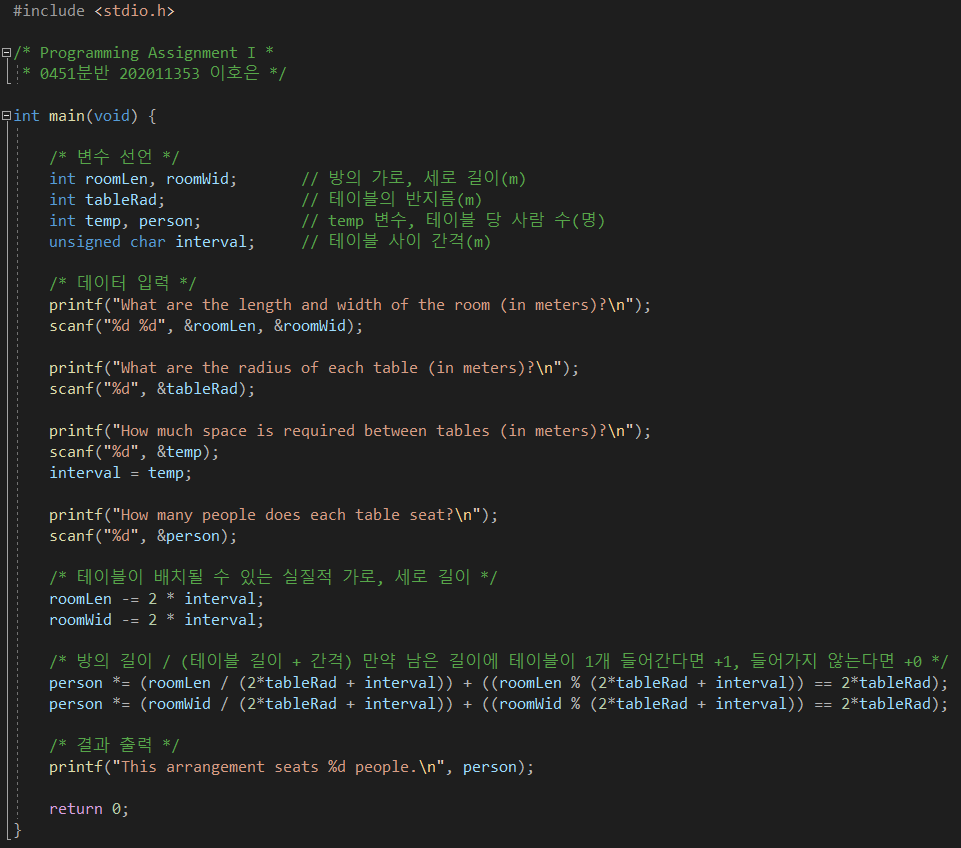
- 여기서 문제가 각 마지막 줄에 배치되는 테이블의 경우 간격을 더하게 되면 끝쪽에는 방으로부터의 간격과 테이블로부터의 간격, 즉 간격이 2개가 생기게 된다. 즉 남은 공간 (방의 길이 / (테이블 지름 길이 + 간격))의 나머지 즉, (방의 길이 % (테이블의 지름 길이 + 간격)) 내에 테이블이 들어갈 수 있는지를 확인해야 한다.

- 이때, 테이블의 길이보다 나머지가 크다면 몫이 +1이 될 것이므로 나머지의 최댓값은 테이블의 길이와 같다. 즉, 나머지 공간에 테이블이 한 줄 더 들어갈 수 있는 경우는 (나머지 = (방의 길이 % (테이블의 지름 길이 + 간격)) == 테이블의 지름 길이)가 될 것이다. 이때, 만약 나머지 공간에 테이블이 들어갈 수 있다면 조건식이 참이 되어 +1이 될 것이고 테이블이 들어갈 수 없다면 조건식이 거짓이 되어 +0이 될 것이다.

- 알고리즘

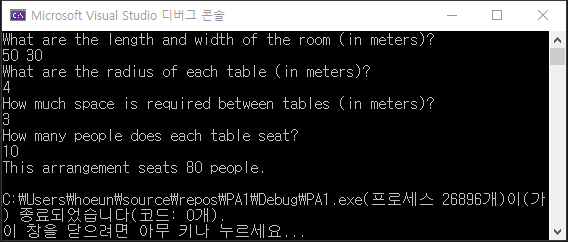
|  |
| --- |
| ⓐ 데이터를 입력 받는다.  ⓑ 실제로 테이블이 배치될 수 있는 방의 가로, 세로 길이를 각각 구한다. = (방 길이 – 2 \* 간격)  ⓒ 실질적 방 가로 길이 / (테이블 지름 길이 + 간격)  ⓓ 만약 ⓒ에서 테이블이 배치되고 남은 길이에 테이블이 들어갈 수 있다면 +1  ⓔ 이를 person 변수의 값과 곱하여 person 변수에 대입한다. (사람수 x 가로 개수)  ⓕ 실질적 방 세로 길이 / (테이블 지름 길이 + 간격)  ⓖ 만약 ⓕ에서 테이블이 배치되고 남은 길이에 테이블이 들어갈 수 있다면 +1  ⓗ 이를 person 변수의 값과 곱하여 person 변수에 대입한다. (사람수 x 가로 개수 x 세로 개수)  ⓘ 결과를 출력한다. |

4) 소스코드



5) 수행 결과

- 주어진 Sample Data를 이용한 수행



- 데이터 셋을 통한 프로그램 검증

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 데이터 설정 | | | | |
| int roomLen = 1000, roomWid = 1000  int tableRad = 1  int interval = 1, person = 20 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | | | 2,217,780명 |
| 검증 | | | | |
| (최대의 경우)  1000m x 1000m의 방 안에 1m x 1m의 테이블이 1m 간격으로 배치되면 이와 같은 모습이다.  가로로 테이블은 간격, 테이블, 간격, 테이블, …, 테이블, 간격 순으로 놓이게 된다. 1000m일 때 1m의 간격을 두고 지름 2m 길이의 테이블은 332개가 들어갈 수 있다. (998 / 3)  이때 332개를 배치하고 남은 공간이 2m가 남으므로 테이블이 1줄씩 더 배치될 수 있다. 즉 332 + 1개씩이다.  가로 333개 x 세로 333개 = 110,889개이다.  110,889개 x 20명 = 2,217,280명이다. | | | | |
| 2 | 데이터 설정 | | | | |
| int roomLen = 100, roomWid = 100  int tableRad = 40  int interval = 10, person = 1 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | | | 1명 |
| 검증 | | | | |
| (정확히 일치하는 경우)  100m x 100m의 방 안에 80m x 80m 테이블이 10m 간격으로 배치되면 이와 같은 모습이다.  실제로 테이블이 배치될 수 있는 공간의 크기는 가로, 세로 길이가 100m – 2 \* 10m = 80m이다.  80m x 80m의 공간 안에 80m x 80m 크기의 1개 테이블이 들어갈 수 있다. (정확히 일치)  1개의 테이블에 1명이 앉는다면 1명이다. | | | | |
| 3 | 데이터 설정 | | | | |
| int roomLen = 100, roomWid = 100  int tableRad = 110  int interval = 1, person = 1 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | | 0명 | |
| 검증 | | | | |
| (초과의 경우)  100m x 100m 방에 220m x 220m 크기의 테이블이 들어갈 수 없으므로 0명이다. | | | | |
| 4 | 데이터 설정 | | | | |
| int roomLen = 100, roomWid = 100  int tableRad = 25  int interval = 15, person = 1 | | | | |
| 프로그램 출력 결과 | | | | |
|  | 1명 | | | |
| 검증 | | | | |
| 100m x 100m의 방 안에 50m x 50m 테이블이 15m 간격으로 배치되면 이와 같은 모습이다.  실제로 테이블이 배치될 수 있는 공간의 크기는 가로, 세로 길이가 100m – 2 \* 15m = 70m이다.  70m x 70m의 공간 안에 50m x 50m 크기의 1개 테이블이 들어갈 수 있다. (남은 공간의 가로, 세로 길이는 20m 이므로 테이블이 들어갈 수 없다.)  1개의 테이블에 1명이 앉는다면 1명이다. | | | | |
| 5 | 데이터 설정 | | | | |
| int roomLen = 200, roomWid = 200  int tableRad = 25  int interval = 20, person = 10 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | 100명 | | |
| 검증 | | | | |
| 가로의 경우 160m(200m – 2\*20m) 안에 지름 50m 길이의 테이블이 20m의 간격을 두고 2개가 들어갈 수 있다. (160 / (50 + 20)) 나머지는 20m이므로 50m의 테이블이 들어갈 수 없다.  세로의 경우 360m(300m – 2\*20m) 안에 지름 50m 길이의 테이블이 20m의 간격을 두고 5개가 들어갈 수 있다. (360 / (50 + 20)) 나머지는 10m이므로 50m의 테이블이 들어갈 수 없다.  가로 2개 x 세로 5개 x 테이블 당 10명 = 100명이다. | | | | |

6) 토의 사항

- 비슷한 유형의 문제는 알고리즘을 잘 개발한다면 여러 경우에 변형하여 사용할 수 있다는 것을 배웠다.

- 1번 문제와 알고리즘이 거의 동일하다. 문제가 요구하는 것이 같다면 알고리즘도 같을 수 있다.

- 이 경우도 많은 코드 수정과 디버깅을 통해 코드를 완성하였다. 수학적인 원리가 적용되는 코드는 알고리즘 개발이 매우 중요하다는 것을 배웠다.

**Question 3. 기차 충돌 시간 계산 문제**

1) 문제 정의

양방향의 철로에서 두 대의 기차가 마주보며 달려오고 있다. 두 기차 사이의 거리, 각 기차의 이동속도를 입력 받고, 충돌까지 남은 총 시간을 분과 시, 분, 초 단위 두 가지로 출력하고, 각 기차의 이동 거리를 출력한다.

|  |  |
| --- | --- |
| 입력 조건 | 출력 조건 |
| - 기차간 거리는 킬로미터 단위(km)로 양의 정수이다.  - 기차의 이동 속도는 시간당 킬로미터 단위 (km/h)로 양의 정수이다.  - 변수 선언은 입력 조건에 맞는 자료형을 사용한다. - 사용자의 입력 값이 입력 조건을 만족하는지 검사할 필요는 없으며, 입력 조건에 맞는 입력 값이 주어진다고 가정한다. | - 입력 값에 따라, 충돌까지의 시간과 각 기차의 이동 거리를 출력한다.  - 분 단위의 시간은 소수점 이하 3자리까지만 출력한다.  - 시, 분, 초 단위의 시간은 모두 정수이다.  - 이동 거리는 킬로미터 단위로 소수점 이하 2자리까지만 출력한다. |

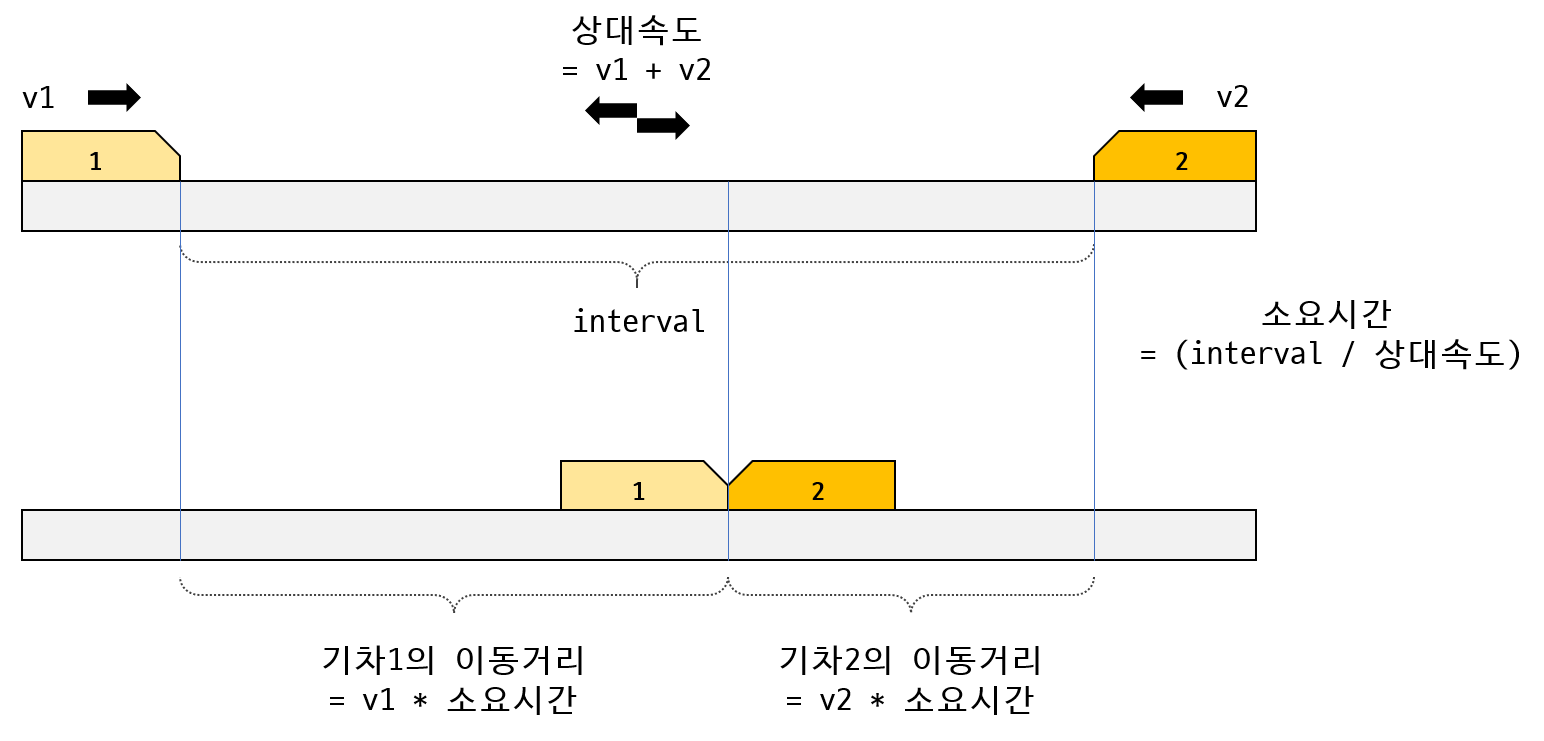
2) 주요 변수 설명

|  |  |  |
| --- | --- | --- |
| 변수명 | 자료형 | 설명 |
| **interval** | int | 두 기차 사이의 간격을 저장한다. (km) |
| **v1** | int | 첫 번째 기차의 이동속도를 저장한다. (km/h) |
| **v2** | int | 두 번째 기차의 이동속도를 저장한다. (km/h) |
| **t** | double | 충돌까지의 소요 시간을 저장한다. (초 단위) |
| **h** | double | 충돌까지의 소요 시간(t)을 시, 분, 초로 변경하였을 때의 시간을 저장한다. (hour) |
| **m** | unsigned char | 충돌까지의 소요 시간(t)을 시, 분, 초로 변경하였을 때의 분을 저장한다. (min) |
| **s** | unsigned char | 충돌까지의 소요 시간(t)을 시, 분, 초로 변경하였을 때의 초를 저장한다. (sec) |
| **MINTOSEC** | const int | 1분을 초 단위로 변경하였을 때의 값을 저장한다. |
| **HOURTOSEC** | const int | 1시간을 초 단위로 변경하였을 때의 값을 저장한다. |

- MINTOSEC의 값인 60과 HOURTOSEC의 값인 3600은 정해진 값이므로 기호상수로 정의해주었다.

- 0~60의 범위 내 정수를 값으로 가지는 m(분 단위)과 s(초 단위) 변수는 unsigned char 타입으로 선언해주었다.

3) 아이디어 및 알고리즘



- 두 대상이 서로 반대 방향으로 이동하고 있을 때, 전체 거리를 이동하는데 소요한 시간을 구하고자 한다면 두 대상의 상대속도를 구한 후, 거리를 상대속도로 나누어야 한다. 상대속도는 두 기차가 반대방향으로 이동하므로 v1 - (-v2) = v1 + v2이며, 거리는 두 기차 사이의 거리이다. 즉, 총 소요시간은 (거리/속도)인 (interval / (v1 + v2))가 된다.

- 세 변수에는 시간단위인 km/h, 그리고 이에 상응하는 km 단위의 값이 들어가 있기 때문에 연산을 할 경우 소요 시간의 단위는 시가 된다. 이때, (interval / (v1 + v2))는 정수형 변수 피연산자들로 구성된 연산식이므로 결과도 정수로 출력된다. 이렇게 될 경우 시 단위의 소요 시간에서 소수점 이하가 절삭된다면 분, 초로 변환하였을 때 절삭오류가 발생할 수 있다. 즉, 세 변수를 double형으로 캐스팅하여 결과를 double로 반환해주어야 한다.

- 위 연산에서 계산한 값은 시 단위의 값이므로 가장 작은 단위인 초 단위로 변환해주기 위해 HOURTOSEC(3600)을 곱한다. (작은 단위로 계산해야 나눗셈과 모듈러 연산(mod)을 통해 큰 단위의 값으로 변환하기 쉬워진다.)

- 가장 큰 단위는 시단위 이므로 초 단위의 t를 HOURTOSEC으로 나누어준다. 이때, t의 자료형을 임시적으로 캐스팅 연산을 통해 정수형(int)으로 변환해준다. (캐스팅 연산은 변수 자체의 자료형을 변환하는 것이 아니라 임시적으로 변환하는 것이기 때문에 t에 저장된 값 자체가 절삭되거나 변하지는 않는다.) 피연산자를 전부 정수로 만들어 결과가 정수로 반환되게 한다. 시간은 정수값을 취하기 때문이며, 나머지를 분, 초로 표시하여야 하기 때문이다.

- 위와 같은 원리로 분, 초를 구한다. 이때, 초 단위의 소요시간인 t의 값을 HOURTOSEC으로 나눈 나머지 값이 분, 초로 변환해야할 값이 된다. 모듈러 연산(mod)을 이용하여 나머지를 구해준 후 MINTOSEC으로 나누어 분 단위 시간을 구한다. 초 단위는 위와 마찬가지로 HOURTOSEC과 MINTOSEC으로 나눈 후의 나머지가 초 단위 시간이 된다. 이 때에도 마찬가지로, 시간은 정수값을 취하므로 형변환연산을 통해 피연산자를 정수형으로 캐스팅하여 결과가 정수형으로 반환되도록 한다.

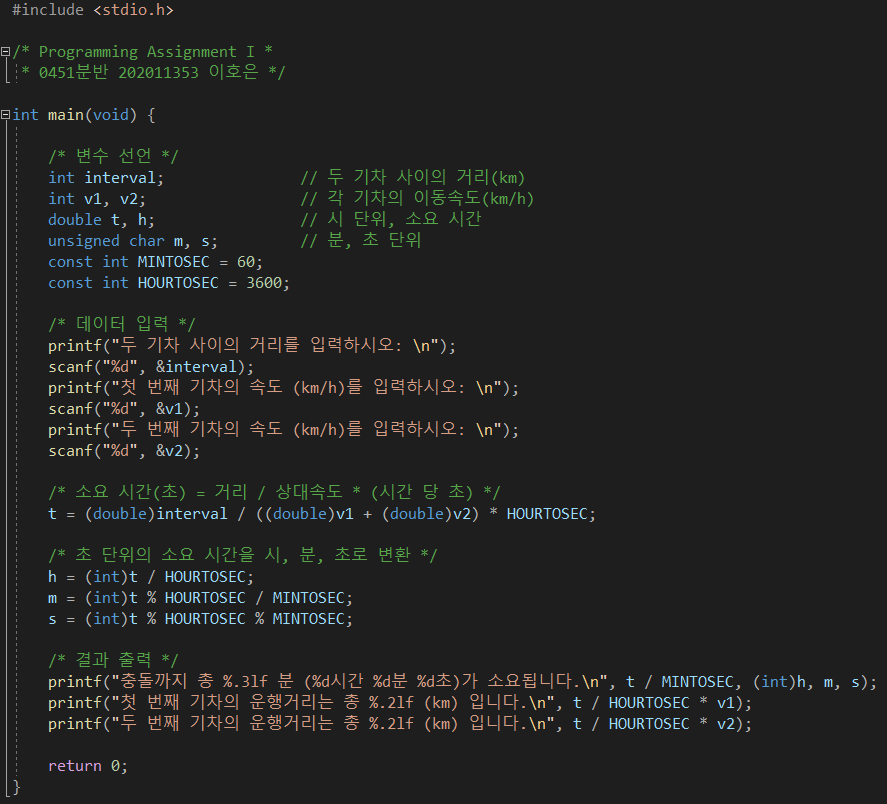
- 각 기차의 이동거리는 (거리 = 속력 x 시간)이므로 각 기차의 속력(v1과 v2)의 총 소요시간을 곱해주면 각 기차의 이동거리를 계산할 수 있다. 이 때, 거리와 각 기차의 속도는 정수형 변수이고 시간은 double형이기 때문에 결과는 double형으로 반환된다. 이때, t는 초단위의 시간이 저장되어 있으므로 HOURTOSEC으로 나누어 시간단위로 변환해주어야 한다.

- 이동 거리는 소수점 이하 2자리, 분 단위의 시간은 소수점 이하 3자리로 출력되므로 %.2lf, %.3lf와 같은 적절한 형식지정자를 이용한다.

- 알고리즘

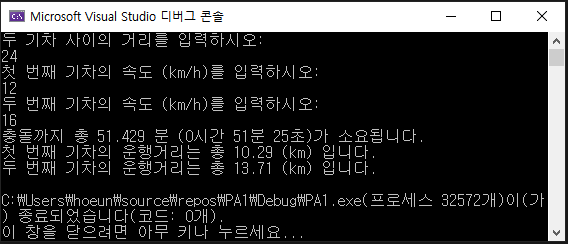
|  |
| --- |
| ⓐ 각 변수의 값을 scanf() 함수를 이용해 입력 받는다.  ⓑ 초 단위의 소요시간을 구한다. (거리 / 상대속도 \* 시간 당 초)  ⓒ 초 단위의 소요시간을 위에 서술한 연산을 통해 시, 분, 초로 변환하여 각각의 변수에 값을 대입한다.  ⓓ 분 단위의 소요시간을 소수점 이하 3째자리까지 출력한다. 또한, 시, 분, 초로 변환한 시간을 출력한다.  ⓔ 각 기차의 이동거리를 출력한다. (소요시간(시간 단위) \* 각 기차의 속도)  ⓕ 변수에 저장된 총 사람 수를 printf() 함수를 이용해 출력한다.  ⓖ main 함수를 종료한다. (return 0) |

4) 소스코드



5) 수행 결과

- 주어진 Sample Data를 이용한 수행



- 데이터 셋을 통한 프로그램 검증

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 데이터 설정 | | | | |
| int interval = 30  int v1 = 3  int v2 = 5 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | | | 225분, 3시간 45분 0초 소요  운행거리 각각 11.25km, 18.75km |
| 검증 | | | | |
| 두 기차의 상대속도는 3km/h + 5km/h = 8km/h이다. 시간 = 거리 / 속력이므로 30km / 8km/h = 3.75h  즉, 3.75시간이 소요된다. 3.75시간은 225분이다. (3.75 x 60분) 그리고, 225분은 3시간 45분 0초이다.  거리 = 속력 x 시간이므로 기차1은 3.75h x 3km/h = 11.25km를 이동하며  기차2는 3.75h x 5km/h = 18.75km를 이동한다. | | | | |
| 2 | 데이터 설정 | | | | |
| int interval = 100  int v1 = 10  int v2 = 10 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | | | 300분, 5시간 0분 0초 소요  운행거리 각각 50km, 50km |
| 검증 | | | | |
| 두 기차의 상대속도는 10km/h + 10km/h = 20km/h이다. 시간 = 거리 / 속력이므로 100km / 20km/h = 5h  즉, 5시간이 소요된다. 5시간은 300분이다. (5 x 60분) 그리고, 300분은 5시간 0분 0초이다.  거리 = 속력 x 시간이므로 기차1은 5h x 10km/h = 50km를 이동하며  기차2는 5h x 10km/h = 50km를 이동한다. | | | | |
| 3 | 데이터 설정 | | | | |
| int interval = 10  int v1 = 1  int v2 = 2 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | 200분, 3시간 20분 0초 소요  운행거리 각각 3.33km, 6.67km | | |
| 검증 | | | | |
| 두 기차의 상대속도는 1km/h + 2km/h = 3km/h이다. 시간 = 거리 / 속력이므로 10km / 3km/h = 3.33h  즉, 3.33시간이 소요된다. 3.33시간은 200분이다. (3.33 x 60분) 그리고, 200분은 3시간 20분 0초이다.  거리 = 속력 x 시간이므로 기차1은 3.33h x 1km/h = 3.33km를 이동하며  기차2는 3.33h x 2km/h = 6.67km를 이동한다. (소수점 셋째자리에서 반올림하였다.) | | | | |
| 4 | 데이터 설정 | | | | |
| int interval = 0  int v1 = 10  int v2 = 20 | | | | |
| 프로그램 출력 결과 | | | | |
|  | | | 0분, 0시간 0분 0초 소요  운행거리 각각 0km, 0km | |
| 검증 | | | | |
| 두 기차의 상대속도는 10km/h + 20km/h = 30km/h이다. 시간 = 거리 / 속력이므로 0km / 30km/h = 0h  즉, 거리가 0km라면 충돌하는 순간이다. 0시간은 0분이다.  거리 = 속력 x 시간이므로 기차1은 0h x 0km/h = 0km를 이동하며  기차2는 0h x 0km/h = 0km를 이동한다. 즉, 충돌하는 순간인 경우이다. | | | | |
| 5 | 데이터 설정 | | | | |
| int interval = 10  int v1 = 200  int v2 = 350 | | | | |
| 프로그램 출력 결과 | | | | |
|  | 1.091분, 0시간 1분 5초 소요  운행거리 각각 3.64km, 6.36km | | | |
| 검증 | | | | |
| 두 기차의 상대속도는 200km/h + 350km/h = 550km/h이다.  시간 = 거리 / 속력이므로 10km / 550km/h = 0.018h  즉, 0.018시간이 소요된다. 0.018시간은 1.091분이다. (0.018 x 60분) 그리고, 1.091분은 0시간 1분 5초이다.  거리 = 속력 x 시간이므로 기차1은 0.018h x 200km/h = 3.64km를 이동하며  기차2는 0.018h x 350km/h = 6.36km를 이동한다. (소수점 셋째 자리에서 반올림하였다.) | | | | |

6) 토의 사항

- 이 프로그램은 물리학 지식이 필요하다. 두 물체의 이동거리와 이동시간은 상대속도(반대방향이므로 절댓값의 합)를 통해 구한다는 사실이다.

- 프로그램을 짜며 수학적 원리가 기반이 된 물리학적인 문제도 수학기반의 프로그래밍으로 손쉽게 계산할 수 있다는 것

- 소수점 3째자리에서 반올림하였더니 오차가 매우 작아 무시할 수 있는 수준이었다.

- 프로그래밍에서 변수의 메모리 할당 등이 실행 속도에 영향을 미친다는 것을 배웠다.

- 범위가 크지 않은 양의 정수값은 unsigned char형 변수로 사용하면 1바이트의 메모리만 할당 받아 실행속도를 증가시키고, 불필요한 메모리 낭비를 덜 수 있다.