코드없이

배우는

프로그램

저자 : foxman12@hanmail.net

basic-  
01. 프로그램 기초 시작하기

———————————————————————————————————————————————--———

## **프로그램 관련 용어 살펴보기 : 이후 목차 변경 필요**

## **알고리즘과 순서도**

## **순서도 사용방법**

## **변수와 우선순위 개념**

## **여러가지 순서도 만들기**

## **순서도 답안**

## **배열 순서도**

## **블랙잭 순서도**

|  |
| --- |
| > 01. 프로그램 관련 용어 살펴보기 |

프로그램 용어가 특별히 중요해 보이지는 않아도 누군가 본인에게 물어 본다면 답 할 수 있어야 한다. 설명 할 수 없는 것은 모르는 것이다. 라는 유명한 말이 있다. 설명을 하다 보면 본인 스스로 정리가 되어서 전체를 이해하는데 도움이 될것이다. 프로그램관련 용어를 살펴 보자.

프로그램이란 무엇인가?

결과를 얻기위해 정해진 순서대로 진행되는 과정을 프로그램 이라 한다. 예를 들어 우리가 다이어트 프로그램을 따라서 진행하면 살이 빠지게 될것 이고, 요리 만드는 프로그램에 따라 순서대로 진행 한다면 요리가 만들어 질 것이다.

프로그램 실행이란?

특정 목적을 이루기 위한 과정을 상세히 기술한 프로그램을 기술된 순서대로 진행 하는 것을 말한다.

컴퓨터 프로그램이란?

사람이 컴퓨터에게 어떤 특정 작업을 시키고 싶을 때 컴퓨터가 해야 할 일들을 순차적으로 상세히 기술한 결과물이 있어야 되는데 이를 컴퓨터 프로그램이라 한다.

여기서 특정 작업이란? 컴퓨터에서 할수 있는 동영상보기, 음악듣기, 문서 작업 하기,게임하기 등을 의미 한다.

컴퓨터가 컴퓨터 프로그램을 한줄 한줄 순서대로 실행 하면 컴퓨터 사용자가 게임도 할 수 있게 되고, 동영상 편집도 할수 있게 되고, 문서편집도 할 수 있게 된다.

프로그램 언어란 무엇인가?

컴퓨터 프로그램을 만들때 사용하는 기호를 컴퓨터 프로그램 언어라고 한다. 컴퓨터 프로그램을 만들면 프로그램 내용 대로 컴퓨터는 한 줄 한 줄 해석해서 컴퓨터에서 원하는 결과를 얻을 수 있다. 프로그램 언어 명령어 하나 하나가 컴퓨터가 어떻게 하라는 명령어가 된다. 이때 사용하는 대표적인 프로그램 언어로 c언어, java, 파이썬, 자바스크립트 등이 있다.

현실 세계의 언어를 생각해 보면 한국어는 한국사람들이 서로 의사 소통하기 위해서 만든 언어이고 영어는 영국, 미국 사람이 서로 의사 소통하기 위해서 만든 언어이다. 중국어는 중국 사람이 서로 의사 소통을 하기 위해서 만든 언어이다. 컴퓨터에게 동영상 좀 틀어 달라고 아무리 이야기해도 컴퓨터는 알아듣지 못한다. 컴퓨터가 알아 들을 수 있는 언어로 동영상을 틀어 달라고 해야 한다. 컴퓨터가 알아들을 수 있는 언어가 프로그래밍 언어 있다. 프로그램 언어란? 컴퓨터 같은 기계장치와 의사 소통 하기 위해서 만든 언어이다. 우리가 미국사람과 대화를 나누기 위해서 영어를 공부해야 하듯이 컴퓨터와 대화를 나누고 싶다면 프로그램 언어를 배워야 한다. 결국 프로그램은 프로그램언어로 만들어야 컴퓨터 같은 기계장치에서 동작할 수 있다. 프로그램 언어는 현실세계의 언어처럼 하나만 존재하는 것이 아니고 여러개 존재한다. 이 책에서는 많은 프로그램 언어 중 자바를 배우게 된다.

지금까지 프로그램, 컴퓨터 프로그램, 컴퓨터 프로그램 언어의 뜻을 정의해 보았다. 이해 될때까지 여러번 읽어보자.

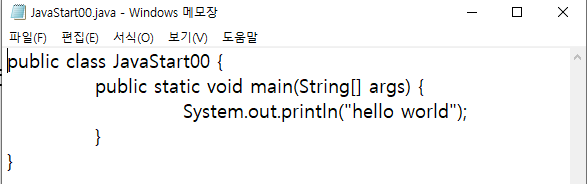
소프트웨어란?

컴퓨터 프로그램 언어로 만들어진 컴퓨터 프로그램으로 모니터 안에서 클릭하면 동작하는 다양한 프로그램들을 의미한다. 즉 컴퓨터 안에서 컴퓨터를 조작해서 사용자가 원하는 결과를 얻어내는 것을 목표로 하는 컴퓨터 프로그램을 의미한다. 여러분들이 날마다 하는 게임, 브라우저, 메신저, 한글, 워드, 엑셀, ppt, 포토샵 등과 같이 컴퓨터 모니터 안에서 실행되고 있는 응용 프로그램을 말한다. 소프트웨어, 응용 프로그램, 컴퓨터 프로그램 다 같은 의미로 보아도 크게 문제가 없다. 실제로 우리가 사용하고 있는 컴퓨터 프로그램을 만들기에는 아직 멀고 험한 길이 기다리고 있지만 참고 견디면 생각보다 쉽게 만들 수 있을 거라 생각된다. 본인의 인생에서 방향 전환은 있을 수 있으나 포기는 있을수 없다. 방향 전환을 잘하려면 가지 않은 길도 잘알아야 가지 않는 이유를 명확히 알수 있다. 본인 이 프로그램 일을 하지 않는다 해도 프로그램 지식은 본인이 살아가는데에 많은 영향을 미칠 것이다. 절대로 프로그램을 포기하지 않기를 바란다. 중간에 그만두는 수많은 학생들을 보면 인생을 편히 살려고 하는 경향이 있다. 고등 학교때 국영수 과목이 인생을 사는데 크게 필요가 없다고 생각한 적이 있다. 지금 생각해 보니 국영수 과목은 도서관에 있는 모든 책을 읽을 때 가장 많이 막히는 부분에 대한 선행 학습 이었던 것 같다. 본인들도 언젠가 사회에서 지식인으로 살아남고자 한다면 언젠간 다시 국영수 과목 책을 펴고 공부 할 날이 올 것이다. 이 책은 프로그램 경험이 없는 사람들에게 경험을 제공해 주는 것을 목표로 한다. 나중에 좀더 복잡한 내용의 자바,c#,파이썬등의 문법책을 공부해 보길 바란다.

코드 란? 컴퓨터 프로그램을 컴퓨터 프로그램 언어로 작성해 놓은 결과물을 말한다. 자바로 예를 든다면 컴퓨터 프로그램을 자바로 구현한 프로그램 결과물을 코드라 한다. 코드 짜는 중이야. 코드좀 만들어줘. 만든 코드를 제출하세요. 등과 같은 용어를 종종 사용한다.

코딩이 란? ‘코드를 작성 하는 중’ 을 의미한다. 코드를 써내려가는 작업을 코딩이라한다. 지금 코딩중이야. 같은 용어를 종종 사용한다.

프로그래머 란? 코드로 프로그램 작성하는 사람을 의미 한다.

왼쪽 이미지는 메모장을 열어서 자바로 프로그램 언어을 기술한 것이다. 이런 것을 ‘코드’라한다. 실제로 옆에 있는 코드는 자바로 만든 코드이고 코드의 내용은 화면에 hello world를 출력하는것이다.

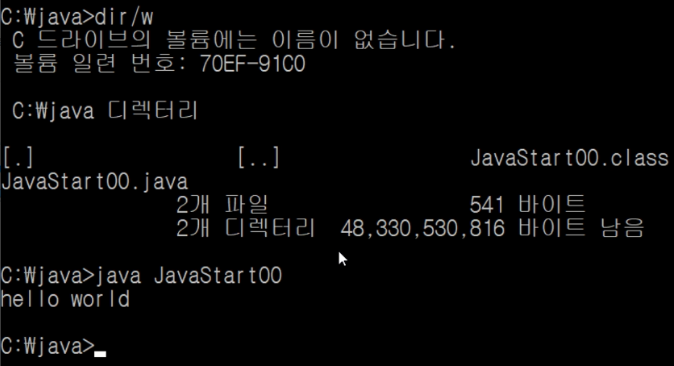
하드웨어 란? 우리 눈에 보이는 모니터 키보드 마우스 본체 등 딱딱한 기계 장치들을 의미한다.

기계장치 란? 시계 핸드폰 컴퓨터 전자렌지 등과 같이 전기를 이용해서 사용자의 요청에 따라 어떤 일을 자동으로 처리해주는 장치들을 의미한다. 이런 기계장치에게 명령을 내리는 것이 프로그램의 목적이다

고급언어 저급언어 란? 프로그램언어는 고급언어와 저급언어로 구성되어 있고 고급 언어는 사람이 알아보기 쉬운 언어이고 저급 언어는 기계장치 컴퓨터가 알아 보기 쉬운 언어이다. 대표적인 고급언어로는 자바 씨샵 씨 등이 있고 저급 언어로는 0과 1로 이루어진 기계어와 어셈블리어가 있다.

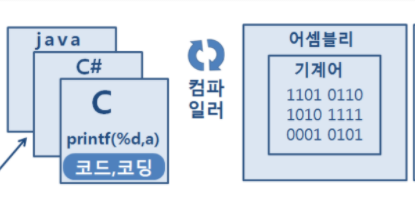
컴파일러 란? 고급 언어를 0과 1로 이루어진 저급언어로 변환해 주는 프로그램을 의미한다 보통 프로그래머가 고급언어로 프로그램을 작성하면 컴파일러가 0과 1로 이루어진 저급언어인 기계어로 변환해준다. 고급 언어는 결국에 0과 1로 이루어진 기계어로 변경되어야 기계장치에서 사용할 수 있다. 계장치에서는 기계어를 이용해서 프로그램을 한줄 한줄 읽어 기계 장치를 동작 시킨다 기계장치는 0과 1만 이해할 수 있기 때문에 컴파일러를 통해서 기계어로 0과 1로 이루어진 저급언어로 변경해 주는 것이다. 저급 언어를 사람이 직접 구현할 수도 있지만 저급 언어는 0과 1로 이루어져 있어 사람이 프로그램 하거나 이해하는 데는 어려움이 있어 저급 언어보다는 고급언어로 작성한다. 따라서, 프로그래머가 알아보기 쉬운 고급언어(자바)로 작성하면 컴파일러가 0과 1로 이루어진 기계어로 변환 한다음 기계어를 기계장치에 넣어 프로그램을 실행 시키다.

아래의 이미지는 상위 이미지의 자바 프로그램을 실행 시킨 결과물이다. 상의 이미지에 나오는 코드는 JavaStart00.java파일에 저장되어 있고 중간에 나오는 JavaStart00.class 파일은 컴파일러라는 프로그램으로 아까 메모장으로 만든 JavaStart00.java파일 문서를 컴파일 하여 0과 1로 이루어진 JavaStart00.class 파일로 만든 것이다.

이미지 아래부분에 hello world라는 문자열이 보일 것이다. java JavaStart00 이라는 명령어를 가지고 JavaStart00.class 을 실행시킨 결과물이다. 

결과적으로 우리가 컴퓨터에서 돌아가는 소프트웨어를 만들려면 메모장으로 프로그램 언어인 자바를 이용하여 코드를 작성한 다음에 컴파일을 통하여 0과 1로 이루어진 기계어로 이루어진 .class 파일을 만든 다음에 .class 파일을 컴퓨터에서 실행시켜 우리가 원하는 결과를 얻는다.

이전 메모장에 기술한 자바 코드는 화면에 hello world를 출력하는 프로그램 코드로 생각하면 되고 다음 검정 바탕에 콘솔 이미지는 프로그래밍을 기계어로 만들어 컴퓨터에서 실행하는 과정이라고 생각하면 된다. 이해하기 어렵다면 일단은 그냥 넘어가도 큰 문제가 없다. 나중에 실제 프로그램언어를 사용해보면 쉽게 이해할 수 있을 것이다.

컴퓨터에 돌아가는 프로그램을 만들려면 코드에서 상의 이미지들 처럼 처음에는 코드를 만들고 코드를 컴파일러를 통해서 0과 1로 이루어진 기계어 파일로 만든 다음, 기계어 파일을 기계 장치에 넣어 프로그램을 실행시키면 기계 장치를 사용할 수 있게 된다.

컴퓨터에게 화면에 hello world좀 출력해줘 라고 하면 컴퓨터가 알아서 화면에 hello world를 출력해 주면 좋겠지만 컴퓨터는 사람의 말을 이해할 수 없고 사람도 컴퓨터의 말을 이해할 수 없다. 00101100101010111 이게 컴퓨터 언어로 화면에 hello world를 출력해 달라는 이야기라면 0010101010111 이것의 의미를 사람이 인식하기는 어렵다. 위쪽 코딩 이미지 처럼 사람이 사용하는 언어는 아니지만 사람이 알아보기 쉬운 프로그램 언어로 작성한 다음 00101100101010111와 같은 기계어로 변환하는 컴파일 과정을 거친다면 누구나 쉽게 컴퓨터 프로그램을 작성 할 수 있을 것이다.

통합 관리 프로그램이란? 이클립스 같은 프로그램 제작을 쉽게 할 수 있도록 도와주는 응용 프로그램 소프트웨어을 의미한다. 통합 관리 프로그램은 고급 언어로 프로그램을 작성하면 자동으로 기계어로 컴파일하여 바로 컴퓨터에서 실행 결과를 볼 수 있게 해주는 소프트웨어이다.

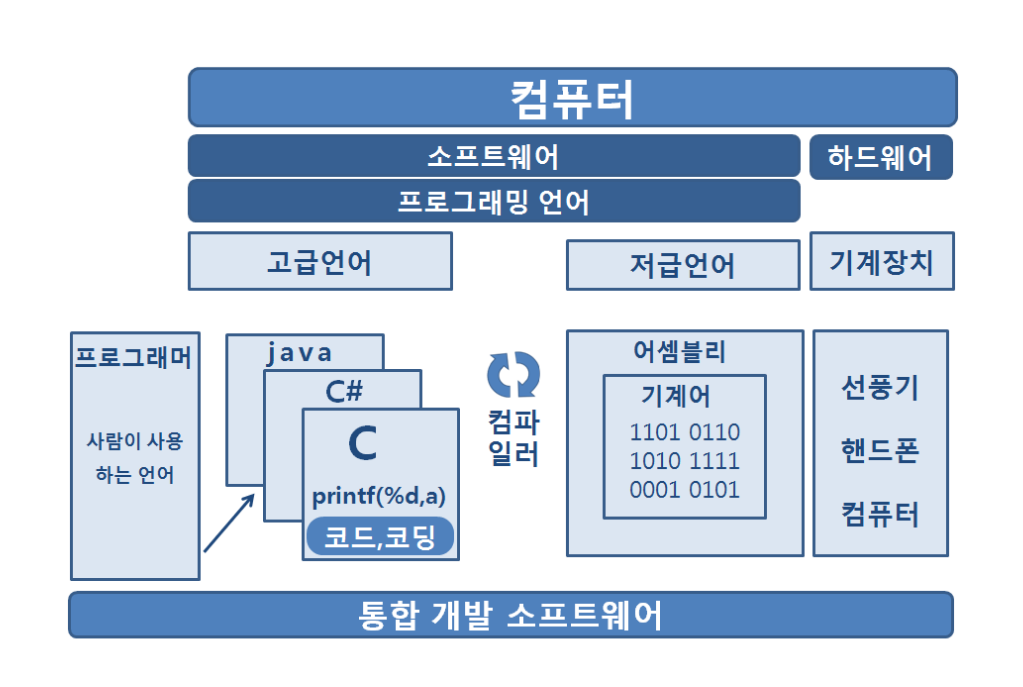
프로그램 만드는 과정을 자바라는 프로그램 언어로 예를 든다면 다음과 같다.

첫번째, .java파일로 프로그래머가 메모장에 자바로 코딩하여 만든다.

두번째, 자바 컴파일러인 javac.exe 파일로 .java 파일을 컴파일하여 이진파일로 된 기계어 .class 파일을 만드는 작업을 한다.

세번째, 컴파일 결과 생성된 .class 기계어 파일을 java.exe 파일을 사용하여 실행시켜 실행 결과를 얻는다.

소프트웨어,컴퓨터 등이 어떻게 구성되어 있고 어떤 과정을 거쳐서 우리가 사용할 수 있게 되는지 표시한 그림이 다음에 있는 그림과 같다. 그림을 확인하면서 앞에 기술된 내용들을 다시 한번 이해 될때 까지 읽어 보자.



컴퓨터는 크게 소프트웨어와 하드웨어로 나누어져 있고 하드웨어는 키보드 마우스 모니터 같은 기계장치로 구성되어 있고 소프트웨어는 프로그래밍 언어로 만들어져 있으며 크게 고급 언어와 저급 언어로 구성되어 있다. 저급 언어는 사람이 알아 보기 어려운 0과 1로 이루어진 어셈블리, 기계어이며 고급 언어는 사람이 알아보기 쉬운 언어로 자바 C# 파이썬 자바스크립트 등이 있다. 요즘 프로그램들은 사람이 알아보기 쉬운 고급 언어로 대부분 작성 되지만 실질적으로 하드웨어 기계장치에서 동작하게 하려면 저급 언어로 만들어야 한다. 이때 사용하는 것이 컴파일러다. 프로그래머가 고급언어로 작성하면 컴파일러가 사람이 알아보기 쉬운 저급언어 기계어로 바꾸고 기계장치가 기계어를 한 줄 한 줄 해석해서 사용자가 원하는 결과를 모니터에 보여준다. 이 복잡한 과정을 쉽게 할 수 있도록 도와주는 소프트웨어가 통합 개발 소프트웨어이다.

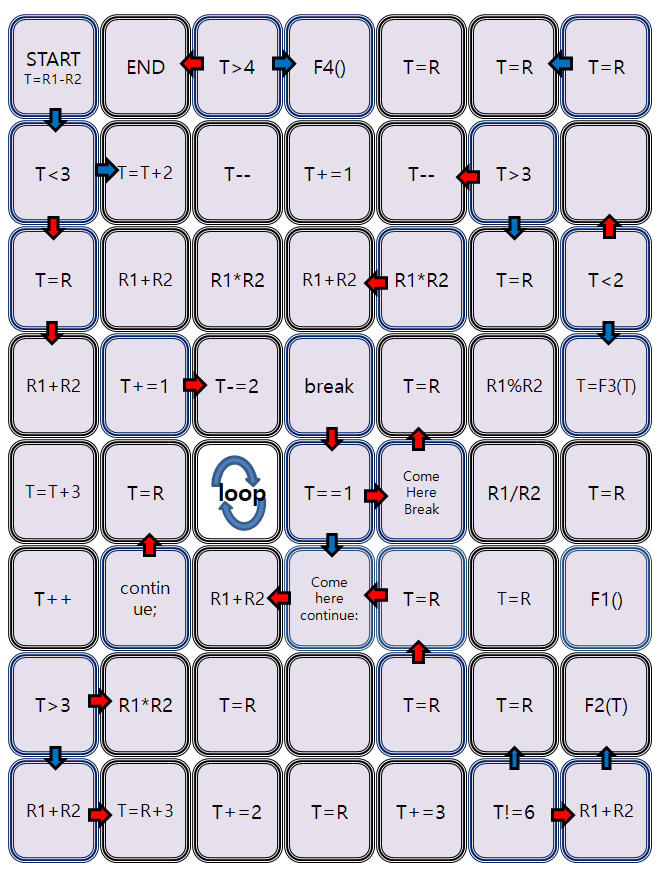
프로그램 언어를 작성해서 컴파일하는 도중에 잘못된 문자가 있으면 문제가 발생 하는데 이런 잘못된 문자를 버그라 하고 잘못된 문자인 버그를 찾는 과정을 디버그,디버깅 이라고 한다.

버그란? 프로그램 언어로 프로그램 작성시 잘못된 프로그램 코드를 의미한다.

디버그, 디버깅이란? 잘못된 코드를 올바르게 수정하는 과정을 의미한다.

컴퓨터의 용도는 결국에는 정보 처리이다. 웹사이트를 생각해보자 정보를 모아 두고 처리해서 사용자가 원하는 형태로 결과를 보여주는 작업을 하는 것이다. 요즘 이런 회사들이 떼돈을 벌고 있다. 구글, 네이버, 다음 등 자세히 보면 정보처리 작업을 하고 있는 것이다. 3+6 에서 3,+,6 이런 것들은 정보에 해당 하고 처리란 실제 계산해서 9라는 결과를 얻는 것을 의미한다. 그래서 컴퓨터 관련 과목들을 정보 처리 과정이라 이야기 하고 해당 자격증 이름을 정보 처리 기사 자격증이라고 한다. 정보처리 기사가 없는 사람이라면 공부해서 꼭 취득 하기 바란다. 2019년 통계에 따르면 모든 직종이 안 좋은 취업률을 보이고 있으나 프로그램 쪽은 올라가고 있다. 미래에는 모든 직종은 사라지고 프로그래머만 남게 될 것이고 그 날이 멀지 않았다. 은행에서 이제는 프로그래머로만 구성된 지점을 선보였다. 인건비가 올라가면서 지점에 모든 사람이 프로그래머로 교체되고 있다. 포기하지 말고 끝까지 ‘살아 남으라’는 이야기를 해주고 싶다. 행운을 빈다. 중간에 모르는 것이 있으면 웹에서 검색해 보자. stack overflow 나 많은 한국사이트가 나올 것 이다. 구글, 네이버, 다음 등과 같은 검색 사이트를 검색 하는 습관을 가지자. 잭 안드리카라는 소년은 15살에 인터넷 검색으로 췌장암 조기진단방법을 찾아냈다. 인터넷을 찾아 집에서 핵융합을 한 사람도 있다. 영어 공부도 열심히 하자. 이 과정이 마무리되면 c언어, java, 파이썬 같은 프로그래밍 기초 언어를 공부해보고 해당 언어로 자료구조를 구현하는 방법을 배워 보도록 하자. 이 2가지만 잘해도 전공자 보다 낫다는 이야기를 들을 것이다.

|  |
| --- |
| > 05. 기차 게임 |

프로그램을 설명할때 많은 책에서 기찻길을 이야기한다. 기찻길 게임을 해보자 기차게임은 왼쪽 이미지 처럼 사각형으로 만들어진 게임 판의 왼쪽 상단 Start에서 시작 한 주사위 말을 각각의 플레이어들이 번갈아 주사의 2개를 던져 나온 수 만큼 화살표 방향을 따라 이동 하다가 왼쪽 상단 END부분에 먼저 도착한 사람이 이기는 게임이다.

게임 판은 왼쪽 이미지가 첫번째 판이고 다음 나오는 이미지는 두번째 판이여서 총 2개의 판으로 이루어져 있다.

말을 이동하는 방법

주사위 하나를 말로 사용하고 게임하는 사람이 돌아 가면서 다른 주사위 2개를 던져 나온 수의 차 만큼 이동 한다. 주사의 2개가 1,6이 나오면 (6-1) 하여 이동 칸수는 5가 되고 주사위가 3,3이 나오면 (3-3)이 되어 이동 칸수는 0이 되어 말은 이동할 수 없다. 주사위가 2,5 이면 (5-2)가 되어 이동 칸수는 3이 된다. 큰수에서 작은 수를 빼면 된다. 예외의 경우가 있는데 만약에 말이 서있는 바닥에 R1+R2, R1\*R2등 R 2개로 만든 수식이 있을 경우 R1은 하나의 주사의 R2는 또 다른 주사위로 생각하여 두 주사위 값을 빼서 이동하는 것이 아니라 바닥에 써있는 계산데로 이동하면 된다. 예를 들어 주사위를 던저 4,3 이 나오면 평상시에는 4-3이되어 1칸 이동 하지만 바닥에 R1+R2가 있을 경우 7칸, R1\*R2가 있을 경우 12칸 이동하면 된다. 연산자를 살펴 보면 + 더하기, - 빼기, \* 곱하기, / 나누기, % 나머지 연산이 있다. 나머지 연산이란 7를 3으로 나누면 몫은 2가되고 나머지는1이 된다. 7%3의 결과는 7를 3으로 나눈 나머지인 1이 된다. 소수점이 없는 숫자를 정수라 하고 소수점이 있는 숫자를 실수라 한다. 주사위 게임에서는 정수 연산을 하여서 계산 결과는 정수가 된다 따라서, 7/3은 2.3333… 이 아닌 2가 된다. 실제 자바와 같은 몇몇 프로그램도 정수의 계산결과는 정수가 되고 실수의 계산결과는 실수가 된다.

T에 의미

말판을 보면 T가 있는데 말이 이동할때 가지고 이동하는 값이다. 주사위 하나를 말로 사용 한다고 이야기 하였다. T에 의미는 이동할 때 사용하는 말이 가지고 있는 숫자인데 말 자체도 주사위여서 이동시 1부터 6사이 숫자중 한 개의 숫자를 사람들이 확인 할 수 있는 상단에 표시한 채 이동 할 수 있는데 이동중 상단에 표시한 주사위 말의 값을 T값이라 한다.

T값이 변경

주사위 말 T값을 변경하고 싶으면 T의 값을 변경할 수 있는 칸에 말이 멈춰선 후, 다음 차례가 되어 주사위 2개를 던져서 나온 수를 가지고 말이 멈춰섰던 칸에 기술 되어 있는 대로 T값을 연산 하여 T값을 변경한후 이동하면 T값을 변경할 수 있다. 해당 칸은 다음과 같다. T=R1-R2, T=R, T++,T+=1,T--등이 기술 되어 있는 칸 이다. T=R1-R2에 의미는 주사위 2개를 던졌을 때 R1 하나의 주사위에서 다른 주사위 R2값을 뺀 값을 T에 넣으라는 의미다 예를 들면 6과 3의 나오면 6에서 3을 뺀 3이 T값이 된다. 게임을 처음 시작해서 말이 처음 출발할 때 start 부분에 말이 놓여져 있고, 해당 블록에 다음과 같은 T=R1-R2 라는 수식이 있다. 따라서 말이 start부분에서 출발하게 되면 그때 비로소 T값이 정해진다. T값을 변경 하는 다른 수식을 살펴 보자. T=R 의 경우는 두 개의 주사위 중 1개의 값을 T에 넣으라는 의미이다. 예를 들어 주사위가 2와 4가 나오면 T 값은 2와4 중 본인이 원하는 값으로 변경하면 된다, T++ 인 경우에는 기존 T 값에서 하나를 증가 시키라는 의미이다 예를 들어 T값이 현재 2이면 3이 되고 5이면 6이 된다. T+=1 T=T+1 둘 다 T의 값을 하나 증가시키라는 의미이다 ,T--은 T=T-1, T-=1과 같은 의미로 T값을 하나 감소 시키라는 의미이다. T값의 변경 시점은 말이 멈췄다가 출발하는 시점이 된다.

T 값이 1~6사이 숫자가 아닐경우

T 값이 변경될 때 1 부터 6 사이의 숫자가 아닐 경우 6보다 큰 경우와 1보다 작은 경우가 있는데 두 상황에 따라 다르게 처리한다.

T값이 6보다 큰 경우 T 값을 6으로 나눈 나머지를 T값으로 한다. 예를 들어 T 값이 7이면 7를 6으로 나눈 나머지는 1이며 T값은 1이 된다. 수학적 기호로 다음과 같이 표시한다 7%6 == 1, 8%6==2가 된다. T값이 8이면 결국에 T값은 2로 변경된다. 9를 6으로 나눈 몫은 1이고 나머지는 3이다 따라서, 9%6==3이 된다.

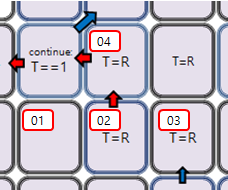
T값이 1보다 작은 경우 T값에 6을 더한 값이 T값이 된다. 예를 들어 T값이 0이라면 0+6를 하여 6이된다. T값이 -2라면 -2+6을 하여 T값은 4가 된다.

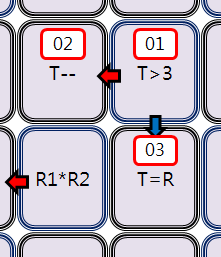
T 값의 용도와 말의 이동방향

T 값의 용도는 말의 이동방향을 결정하는데 사용된다. 판 안에 있는 사각형 색깔을 잘 살펴보면 검정색과 파란색이 있다. 흑백으로 인쇄 되어서 파란색이 약간 흐리고 검정색이 진하게 인쇄되어 있을 것이다. 약간 흐리게 인쇄된 파란색 칸들은 말이 이동 방향을 변경 할 수 있는 곳이다. 파란색 칸은 화살표 꼬리부분이 있는 칸으로 방향 전환이 일어 날 수 있는 칸이다. <-이 화살표 라고 할 때 < 부분이 머리 부분에 해당 하고 - 부분이 꼬리 부분에 해당한다. 말은 꼬리 방향에서 머리 방향으로 움직인다.

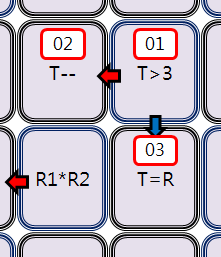
T값을 변경 할 때는 말이 이동한 모든 칸에 있는 T와 관련된 수식들을 적용하는 것이 아니라 말이 멈췄다가 출발할 때 멈췄던 칸의 T관련 수식만을 적용 하여 움직인다. 하지만, 방향 전환과 관련된 파란색 칸은 해당 칸에 멈추지 않더라도 이동 중에 해당 칸을 지나가기만 하더라도 해당 칸에 있는 화살표 방향으로 말의 이동 방향을 변경해 이동 한다.

검정색과 같이 화살표가 전혀 없는 칸을 말이 이동할 때에는 이전 칸에서 말이 이동하던 방향으로 계속해서 말을 이동 하면된다. 결과적으로 화살표 머리부분은 말의 이동에 영향을 주지 않지만 꼬리 부분이 걸쳐 있는 파란색 칸은 말의 이동 방향에 영향을 주어 화살표 방향으로 말의 이동 방향을 변경하게 해준다. 다음 예를 살펴보자.

왼쪽 그림에서 말의 이동을 확인해 보자. 말이 01에서 02로 이동하고 있을 때 01에 멈춰 있는 상황에서 말을 2칸 이동 시켜야 한다고 생각해 보자. 02에 화살표의 꼬리 부분이 없다면 말은 03으로 이동해야 하겠지만 02에 화살표의 꼬리 부분이 있어서 방향 전향이 일어나서 위쪽으로 방향을 바꿔서 04에서 말이 멈추게 될 것이다. 04번 째에서 한칸 더 이동한다면 이동한 말의 위치는 04의 왼쪽이면서 01의 위쪽일 것이다. 

파란칸 중 몇몇은 왼쪽 이미지의 01 처럼 빨간색과 파란색 화살표가 2개 있을 경우가 있는데 해당 칸은 >,<,==과 같은 비교연산자가 해당칸에 존재 하고 비교 연산 결과에 따라 이동 방향이 결정 된다. 비교 연산자 연산 결과가 true이면 말은 빨간색 화살표 방향으로 이동하고, false이면 파란색 화살표 방향으로 이동한다. 말이 이동 방향을 결정하는 파란색 칸을 이동할 때에는 멈춰 있지 않더라도 비교 연산자의 결과에 따라 이동 방향이 변경되어 이동을 계속하게 된다.

비교연산자에 대해 좀더 살펴보면 3 > 4 에서 >는 비교 연산자이고 두수를 비교한 결과가 ‘3이 4보다 크다’ 라는 이야기이고 이는 실질적으로 잘못된 이야기이므로 3>4의 계산 결과는 거짓, false 또는 0이 된다. 3 < 4 의 경우 ‘3이 4보다 작다’ 라는 이야기이고 실제로 3<4 은 3이 4보다 작으므로 참,true,1이 된다. <,==와 같은 모든 비교 연산자의 연산 결과는 true,false 값을 가지는 불리언(boolean) 이라는 자료형이 된다. 자료형이란? 컴퓨터에서 사용할 수 있는 데이터의 형태로 소수점이 없는 숫자인 정수, 소수점이 있는 숫자는 실수, true,false값을 가지는 불리언 문자나 문자열을 나타내는 char나 String 자료형등이 있다.

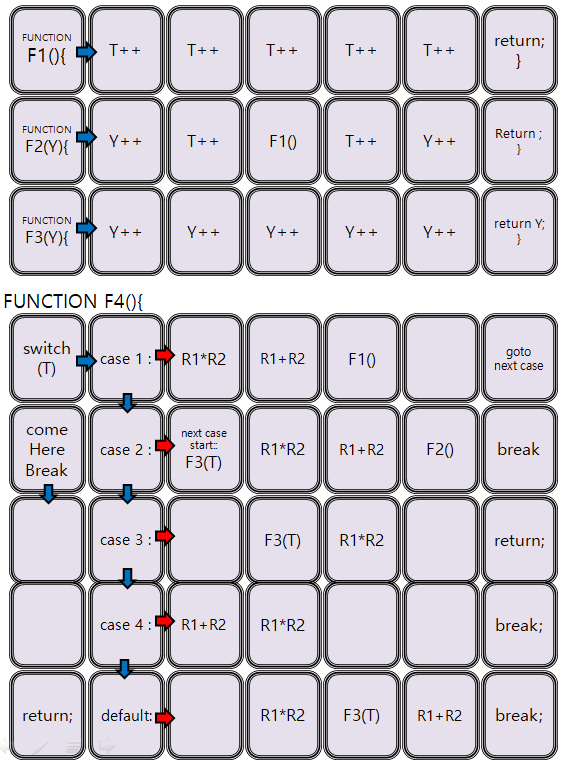
예를 들어 보면 왼쪽 이미지의 01에서 처럼 화살표의 꼬리가 2개인 경우를 만난다면 상황에 따라 02나 03으로 이동할 수 있는데 이때 기준이 되는 것이 T값을 가지고 비교 연산한 결과이다. T값은 현재 이 위치를 지나가는 주사의 말이 위 쪽으로 표시하고 있는 숫자에 해당한다. 판에 쓰여져 있는 형태의 비교 연산식을 T 값을 가지고 비교 연산한 후 참이 나오면 빨간 화살표 방향으로(03) 거짓이 나오면 파란색 화살표 방향으로(02) 말을 이동 시킨다. 현재 T값이 2라면 T<4(2<4)는 참이 되는 것이고 T>4(2>4) 은 거짓이 되는 것이다. 실행 결과가 true, false 결과가 나올 수 있는 비교 연산자 수식은 <, >, !=, == 등이 있고 이 비교 연산자를 사용하면 true, false 값을 얻을수 있다. 이런 true, false의 결과로 진행 방향을 바꿀수 있는 형태를 프로그램에서는 조건문(IF문) 이라고 한다. 15>23 이라면 15가 23보다 크다는 이야기인데 실제로 23이 더 크니 false가 연산 결과가 된다. 맞냐? 틀리냐? true, false 참, 거짓 전산에서는 참을 1, 거짓을 0 으로 표현하기도 한다. 전부 다 같은 의미이다. 프로그램에서 숫자 1을 true 0를 false로 생각 한다. 이렇게 참 거짓 두 개의 값을 가지는 자료형을 불리언 (boolean) 자료형 이라고 한고 -3,-2,-1,0,1,2,3등 은 자료형이 숫자이다 true, false 자료형이 불리언 이다. 숫자 자료형은 2가지 자료형으로 나눌수 있는데 소수점이 없는 숫자를 정수 자료형이라 이야기하고 소수 점이 있는 숫자를 실수 자료형 이라 이야기 한다. 따라서 3 + 4는 7의 경우에는 정수자료형을 연산 한 것이다. 3>5 의 실행 결과는 false이며 비교 연산자 실행 결과는 불리언 자료형이다. 불자료형 이라고도 한다.

비교 연산자의 실행 결과는 true, false 값을 갖는 불리언(boolean)이고 bool이라고도 한다. 다음 불리언 관련 표를 확인해 보고 다음 문제를 풀어 보자. 5<23, 9!=23, 34>=34, 53<=23, 9==5 정답이 필요 하다면 다시 한번 표를 확인해보고 그래도 이해가 가지 않는다면 웹을 검색해서 공부해 보자.

|  |  |  |
| --- | --- | --- |
| **기 호** | **설명** | **예제** |
| A > B | A가 B보다 크면 true | 3>4:false 3>3:false 4>3:true |
| A >= B | A가 B보다 같거나 크면 true | 3>=4:false 3>=3:true 4>=3:true |
| A < B | A가 B보다 작으면 true | 3<4:true 3<3:false 4<3:false |
| A <= B | A가 B보다 작거나 같으면 true | 3<=4:true 3<=3:true 4<=3:false |
| A==B | A와 B가 같으면 true | 3==4:false 3==3:true |
| A!=B | A와 B가 다르면 true | 3!=4:true 3!=3:false |

주의해야 할 = 과 ==

T=1, T==1 첫번째는 T값을 1로 바꾸라는 이야기이고 2번째는 T가 1인지 아닌지 물어보는 것이여서 처음 것은 결과가 T값이 1로 바뀌게 되어 T값 1이 남지만 2번째 비교연산자는 연산이 되어 결과가 true, false 값을 가지는 불리언 형이 된다. 프로그램에서는 처음 것은 할당연산자이고 두번째 것은 비교 연산자여서 전혀 다른 결과를 가지니 주의하자. 말은 이동시 파란색 칸에서 yes일때는 빨간쪽 화살표로 no일때는 파란색 화살표로 이동 한다고 하였는데, 방향전환이 일어나는 파란색 칸에서는 계산 결과가 불리언 형이 나오는 T==1를 사용 하여야 하고 주사위의 T값을 1로 변경 해야 하는 경우에는 T=1를 사용하여야 한다.

함수를 만났다면

첫번째 판(이전판)에서 다음과 같은 문자가 기술되어 있는 경우가 있다. F1(),F2(T),T=F3(T),F4() 이것 들은 함수의 호출부에 해당하고 함수 호출부 칸에 말이 멈추었을 경우 다음 주사위를 던질때 두 번째 판으로 이동해서 이동 전에 쓰여져 있는 F1,F2,F3,F4 와 같은 두번 째 판(왼쪽판) 위치로 이동하여 이동을 이어 나가면 된다.

함수를 게임에서 예를 들어 생각해 보면 유닛이 동굴안에 들어가면 화면이 바뀌어 동굴 안에 유닛이 돌아 다니게 되고 유닛이 동굴 밖으로 나오면 동굴 들어가기전 동굴 입구로 유닛이 돌아와 있게 된다. 이런 것들을 프로그램에서는 함수 또는 메소드라고 이야기 한다. 각 함수에서 어떤 형태로 말이 이동 하는 지는 다음에서 자세히 다루어 보겠다.

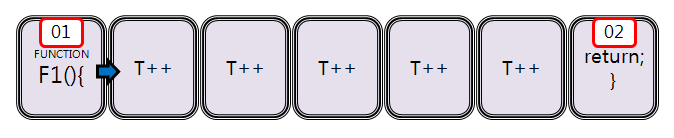
함수 F1()과 F2(Y)의 차이를 설명해 보자면 F1()의 경우 함수를 실행시킬때 필요한 데이터가 없을때 사용하는 것이고 만약 함수를 실행시킬때 필요한 데이터가 Y가 있다면 다음과 같이 F2(Y)로 함수를 호출한다. 이때 Y를 인자, 매개변수라고 한다. 게임에 비유해 본다면 다음과 같다.

F1()의 경우 유닛이 동굴에 들어 갈때 아무것도 들고 들어가지 않는 것을 의미하고, F2(Y)의 경우 유닛이 동굴에 들어갈 때 밧줄 같은 도구를 들고 들어가는 것 처럼 Y를 가지고 들어 간다는 것을 의미한다 전문용어로 매개변수라고 한다.

함수에서 return의 의미를 살펴보면 함수가 종료될때 리턴되는 값을 의미한다. 함수가 실행되고 남은 결과 값을 의미한다. return;의 경우 리턴되는 값이 없다는 이야기이고 return Y;는 리턴되는 값이 Y라는 의미이다. 게임에 비유해 본다면 다음과 같다.

return;의 경우는 유닛이 동굴에서 나올때 아무것도 들고 나오지 않는것을 의미하고 return Y;의 경우에는 유닛이 동굴에서 나올때 Y를 가지고 나온것을 의미한다.

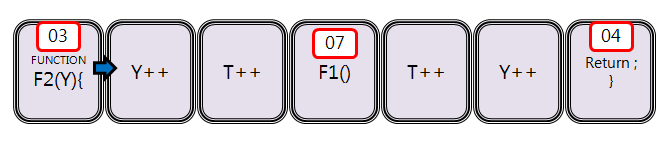
다음은는 F1()함수 이다.



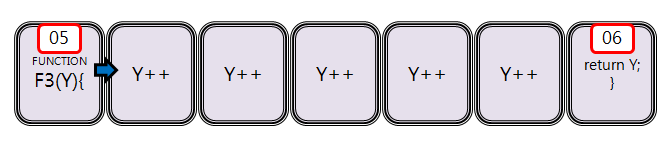
이전 판에서 F1()이라 적혀있는 칸를 만나면 상위 01로 말이 이동한 다음 오른쪽으로 이동하다가 02 return이라고 쓰여져 있는 부분에 도착하면 첫번째 이전 판 F1() 이라고 쓰여져 있는 이전 F1() 부분으로 돌아가 이전 함수에 들어 오기 전의 진행 방향으로 말을 계속 이동 한다. return에 말이 지나가면 걸리지 않더라도 호출된 부분으로 돌아간다.

호출된 부분으로 돌아간다 라는 의미는 이전 판에서 F1()이라고 적혀있는 칸에 말이 걸려 F1()함수로 들어 왔다면 이전 판에 말이 걸린 F1()이라고 적힌 칸으로 돌아 간다는 의미이다.

F1()에서 이동해서 왔을 경우 이전 판의 T값과 F1() 안에 있는 T++ 는 이름만 같은 T이지 다른 T이다. 이동 전 T와 이동한 후 F1() 함수 안에서 T는 아무런 관련이 없다. 따라서, 말이 f1 함수에서 T++ 에 있어도 말의 숫자를 증가시킬 필요가 없다. F1() 함수로 이동 전 T값이 3이면 F1()으로 이동해 다시 복귀한 후에 T값은 F1() 안에서 변경된 T값과 관계 없이 F1()호출전 T값 3값이 유지 된다.



F2(T)를 만나면 F1()과 마찬가지로 F2(Y)라고 쓰여져있는 03으로 이동하여 오른 쪽으로 진행하다가 return 이라고 쓰여져 있는 04를 만나면 첫번 째 판 F2(T)라고 쓰여져 있는 부분으로 복귀하여 이전 이동 방향으로 이동을 계속 한다. 중간 07부분 에 말이 멈추면 함수 F1 으로 이동해 진행을 계속하다가 F1 안에의 return를 만나면 복귀 하여 가던 방향으로 말을 계속 이동 시킨다. F2 함수로 이동 직전 F2(Y){ 라고 되어 있는데 첫번째 판에서의 F2(T)에서 뒤에 소괄호 안의 T값이 F2(Y){의 Y값으로 복사된다. 결론은 F2(Y)안에서 움직일 때 T값을 Y값에 넣어서(복사 해서) 움직 인다는 이야기 이다. 이렇게 되면 03에서 04사이를 움직일 때에 처음에 같은 값을 가지게 되지만 Y++를 만나면 T와 Y는 다르니 Y값만 값이 증가되어 Y++가 실행 되더라도 Y값만 바뀔뿐 T값이 바뀌지 않는다. F2(Y)함수 안에서 T++가 실행 되더라도 주사위의 T 값과 다른 T값이므로 주사위의 T 값은 변경되지 않는다. 첫번째 판에서의 T와 두번째 판 함수안의 T는 다른 T라고 이야기 하였다. 함수가 복귀 되어 첫 번 째 판의 F2(T)위치로 돌아 갔을때 T의 값은 최초 F2(T)함수로 이동 하기전의 T값과 동일해야 정상이다. F2()함수로 이동 전 T값이 3이면 F2() 함수가 복귀한 후에 T값은 F2()함수 안에서 T값과 Y값이 변경 되어 다른 숫자를 가지고 있다 하더라도 복귀후 T값은 영향을 받지 않으므로 T값은 여전히 3이 된다.



T=F3(T)를 만나면 T=F3(Y)로 05로 이동하여 오른 쪽으로 진행하다가 06를 만나면 첫번 째 이전 판 T=F3(T)라고 쓰여져 있는 부분으로 이동 하여 f3함수로 들어오기전 이동 방향으로 이동을 계속한다.

05 부분을 보면 function F3(Y){ 라고 되어 있는데 첫번째 판에서의 T=F3(T)에서 뒤에 T값이 function F3(Y){의 Y값으로 복사된다. T=F3(T)의 경우에는 조금 다른 부분이 있는데 06를 보면 return Y;로 되어 있어 return;의 경우와는 다르다. return;의 경우 아무값도 가지지 않고 복귀 하는데 return Y;일 경우에는 Y 값을 가지고 리턴 된다.

리턴이라는 의미는 호출된 부분으로 돌아가는 것을 의미한다.

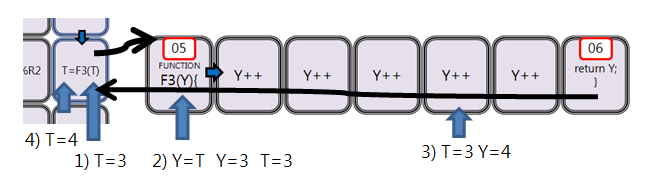
리턴값이란 복귀한 다음 남아있는 값을 의미한다.

결론 부터 이야기 하자면 함수가 복귀 할때 함수 안에서의 Y 값이 T 값에 들어가 T=Y 가 된다.

05에서 06사이의 Y++를 만나면 T값은 증가되지 않고 Y 값만 증가된다. Y값과 T값은 처음에는 같으나 별개의 상태이다. 마지막 부분을 보면 return Y;라고 되어 있다. 말이 06를 지나 함수가 종료되면 첫 번째판으로 이동하여 말이 이동을 계속하는데 복귀 할때 Y 값을 가지고 복귀한다. 주사위 값 T가 Y로(T=Y)로 바뀐다는 이야기이다. Y값을 가지고 리턴하게 되면 T=F3(T)에서 F3(T)부분이 Y값이 되고 T=F3(T) -> T=Y가 되어서 비로소 T값이 Y값이 된다. 이 때 주사위의 값을 Y값으로 바꾼다. F1()의 경우에는 리턴값이 없어서 T=F1()형태의 구문은 사용할 수 없다. T=F3(T)의 경우 F3에서 리턴된 Y값을 T에 넣는 작업을 유닛이 이동 중이라도 실행하고 이동한다. 결론은 함수에서 리턴된 Y가 첫번째 판으로 복귀하면 리턴된 Y로 T값이 바뀐다.

리턴 값의 의미는 F2(T)이 실행되고 (이동한 두번째 판에서 아래 이미지 05에서 시작해서 06에서 끝나면) 남은 값를 의미한다. return Y 라 하면 실행되고 남은 결과가 Y라는 의미이다. F3(T)은 결국 실행된 결과가 return 뒤에 써있는 Y가 되어서 T=F3(T) -> T=Y 과 같이 된다. return;의 경우 리턴값이 없다는 이야기 여서 T=F1() 과 같이 쓰면 T=? 이 되므로 T=F1()과 같이 사용할 수 없다.

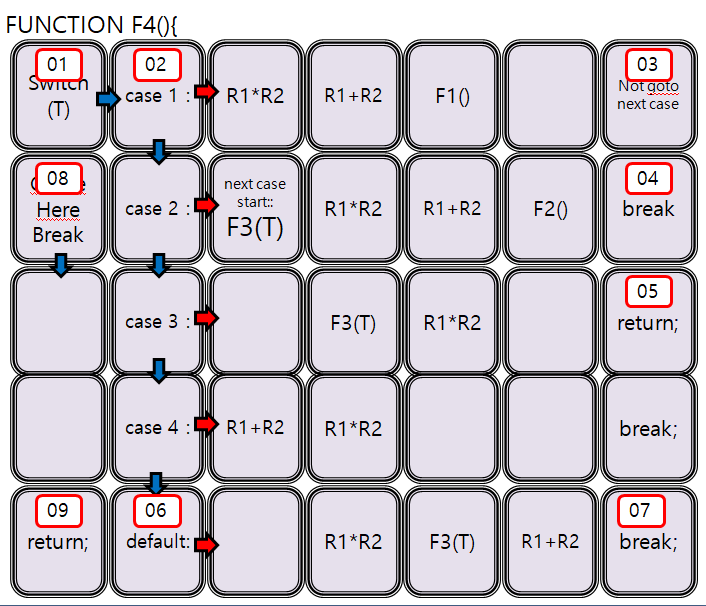
다음 예제를 확인해 보자.



1번 위치에서 T값은 3 2번위치에서는 Y 값 3, T값은 3 3번 위치에서 T값은 3, Y 값은 4 함수 종료시 Y값 4가 리턴되어 T값도 4가된다. T=F3(T)의 경우는 F3에서 리턴된 Y값을 T에 넣는 작업을 유닛이 이동 중이라도 실행하고 이동 한다. 함수를 복귀하면 T값에 리턴값 Y가 반드시 들어간다는 이야기이다.

switch문을 알아 보자

아래 이미지에서 F4()에 들어온 말은 01에서 09방향으로 이동하여 return를 만나서 F4()함수를 빠져나가는 상황인데 01의 switch문을 만나 switch문을 순회하고 08로 복귀하여 F4()함수를 빠져나가게 될 것이다.

왼쪽 이미지에서 FUNCTION F4()안에 SWITCH문을 살펴보면 01에 switch(T) 라고 되어 있다. 말이 02로 이동하면 case 1: 이라고 되어 있는 곳을 지나게 되는데 T값이 1이라면 T값 1과 case 1:의 1를 비교하여 같으므로 true 여서 빨간색 화살표 즉 03쪽으로 이동하게 되어 있다. 만약 T가 1이 아니면 파란색 화살표 아래 쪽으로 이동해서 T와 case 2:를 만나 T값이 2인지 확인하게 될 것이다. 만약에 T값이 2이면 true가 되어서 오른쪽 04 방향으로 이동할 것이고 2가 아니면 case 3: 위치로 이동해서 같은 작업을 계속하다가 결국 06에 도달하게 되면 default:라는 부분에 도착하게되는데 default: 는 T값에 상관없이 모든 경우 실행되게 되어 있다. 이곳에 도착하면 결국 07부분으로 이동하게 되어 있다.

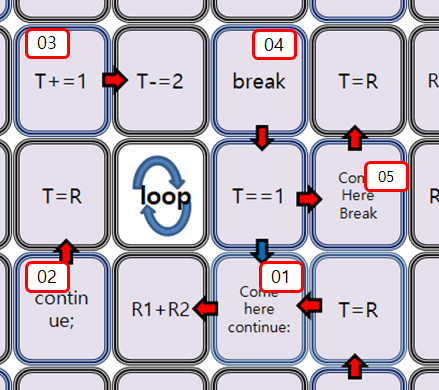
switch문 빠져나가기

switch문에서 적절한 case문을 만나 오른쪽으로 이동하다가 03처럼 break가 없을 경우 다음 case문으로 이동한다. 다음 case문 next case start라고 쓰여져 있는 곳이 있는데 이부분으로 이동할 것이다. 04처럼 break문이 있을경우 switch문을 빠져나와 switch case문의 바깥쪽 부분인 08부분으로 이동하여 나머지 함수 부분을 진행하다가 09를 만나면 함수를 빠져 나가게 된다. 05에 도착 하였을 때는 return를 만나 바로 switch 문을 빠져나갈 뿐만 아니라 함수문도 종료 되어 빠져나가 첫번째 판에서 F4()함수를 호출한 위치로 돌아간다.

switch문의 case문을 나가는 방법은 총 3가지 살펴 봤는데 첫번째는 어떤 키워드도 처리하지 않아서 현재 case문을 빠져나가 다음 case문이 실행되는 방법, 두 번째는 break문을 처리하여switch 문을 빠져나가는 방법, 세 번째는 return 문을 사용하여 switch문 뿐아니라 함수 자체를 빠져 나가는 방법이다.

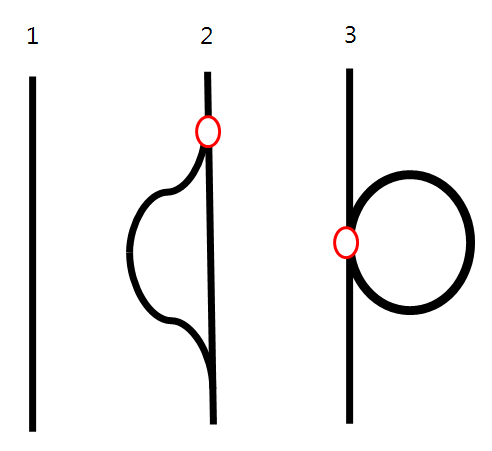
프로그램에서 함수의 목적은 다음과 같다. 함수 특정부분으로 이동하여 작업을 이어서 할때 사용한다. 반복적으로 실행해야 할때 함수를 사용 한다. 함수는 크게 ‘function F1(){}’ 처럼 말이 이동하는 선언부와 ‘F1()’ 처럼 function F1(){} 위치로 이동하는 호출문으로 구성되어 있다. 반복적으로 사용하고 싶을 때마다 ‘F1()’을 기술하면된다. 게임에서 동굴에 들어 갈때마다 동굴 입구에 들어가면 되듯이 프로그램에서는 ‘F1()’을 만날때마다 함수에 들어가 작업을 반복한다.

LOOP 반복문 안에서의 break와 continue

왼쪽 그림에서 1번에 말이 들어 오게 되면 01에서 02, 02에서 03, 03에서 04,04에 다시 01로 무한히 반복되는 말의 이동이 나타난다. 프로그램에서 이런 상태를 반복문이라고 한다. 이렇게 반복되는 부분에서 02 continue를 만나면 반복문이 시작되는 01부분으로 이동하고 04와 같이 break문을 만나면 05로 이동하여 반복문을 빠져 나가게 된다. 결국 break문은 반복문을 빠져나가고, continue는 반복문의 시작 부분으로 이동한다. 이 반복문을 빠져나가기 위해서는 05왼쪽부분의 T==1부분의 조건을 true로 만족하거나 04의 break문에 말이 멈춰야 반복되는 부분을 빠져 나올수 있다. 반복문을 영어로 loop라 이야기하고 반복문을 영원히 빠져나갈 수 없는 경우 무한루프라고 한다. 모든 설명이 끝났다. 다른 사람과 게임을 진행해 보기로 하자. 용어 정리를 해보고 재미있게 여러 사람과 게임을 즐겨보자. 

|  |
| --- |
| > 06. 프로그램의 3요소 |

컴퓨터 프로그램은 크게 3가지로 구성되어 있는데 순차문 반복문 제어문 이다. 이 3요소는 프로그램 실행 순서와 관계가 있다. 실행 순서를 이야기 할 때 철도를 이야기 하는 경우가 많은데 기차가 철도를 따라 순서대로 진행되기 때문이다. 프로그램도 위에서 아래로 순서대로 진행된다. 프로그램 3요소를 철도 그림으로 그리면 아래와 같이 되는데 1번이 순차문 2번이 조건문 3번이 반복문에 해당한다.



**1번 순차문**

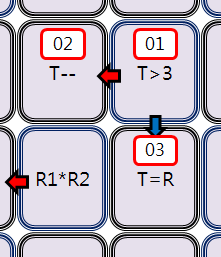
기차가 라인을 따라서 순서대로 움직이듯 실행 코드도 사람이 책을 읽을 때 처럼 순서대로 진행된다.

냉장고에 코끼리 넣는 방법을 3단계로 생각해 보자. 1단계 냉장고 문을 연다 2단계 코끼리 넣는다 3단계 냉장고 문 닫는다. 냉장고 안에 기린 넣는 방법을 4단계로 생각해 보자. 1단계 냉장고 문을 연다 2단계 코끼리를 뺀다 3단계 기린을 넣는다 4단계 냉장고 문을 닫는다. 이전 이야기가 종료된 것이 아니어서 이전 처럼 냉장고 문 열고 기린 넣고 냉장고 문 닫는 방법으로는 해결이 안 된다. 프로그램도 종료되기 전에 실행한 결과가 이후에 계속 영향을 준다. 당연한 이야기 처럼 들리 겠지만 초보자들은 익숙하지 않아서 쉽게 잊어 버리고 코끼리를 꺼내지 않고 기린을 집어 넣으려고 하는 경우가 자주 발생 한다. 냉장고 안에 들어 있는 데이터가 중요도가 있다면 냉장고 안에 무엇이 들어 있는지 항상 파악하고 있어야 한다. 수많은 초보자들이 프로그램 역시 프로그램이 종료되기 전까지 코드들은 서로 연결되어 데이터를 유지 하고 있음을 잊지 말자.

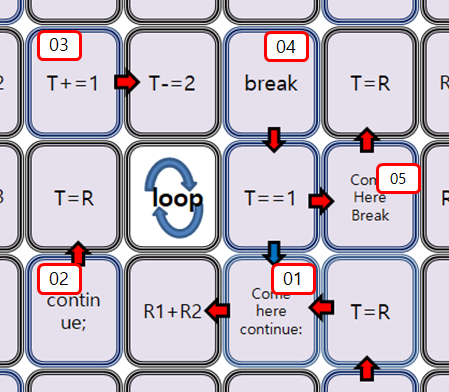
이야기를 이어서 문제를 하나 내 보겠다. 사자가 생일 파티를 열어서 모든 동물한테 반드시 참여 하라고 초대장을 보냈는데 한 동물이 참여하지 않았다. 참여 하지 않은 동물은 어떤 동물일까? 정답은 기린이다. 이유는 냉장고에 갇혀 있어서다. 프로그램이 종료되지 않는 한 지금 해 준 이야기 처럼 계속 연결된다. 프로그램 실행중 가지고 있던 데이터들은 지속적으로 유지, 보관 된다. 초보자들은 프로그램이 순서대로 실행 된다는 것을 알면서도 중간 중간 건너 뛰어서 잘못된 코드 분석을 하는 경우가 많다.

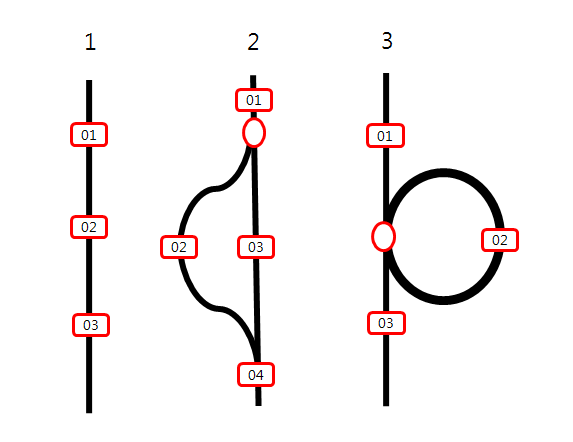
다시 한번 이야기하지만 프로그램은 순서대로 실행이 되고 데이터의 변화도 순서대로 변경되어 프로그램이 종료되기 전까지 유지된다. 아무리 긴 코드를 만나도 당황하지 말고 책을 읽듯이 순서대로 한 줄 한 줄씩 코드를 보자. 프로그램 언어도 결국에는 언어 여서 한 줄 한 줄에 의미가 있다. 따라서, 코드 한 줄 한 줄의 의미를 반드시 이해하고 넘어가야 한다. 한 줄 한 줄 천천히 따라서 일단 보면 책을 읽듯이 언젠가는 코드를 이해 할 수 있을 것이다.

**2번 조건문**

왼쪽 이미지 처럼 둘 중에 하나의 길을 선택해야 할 경우 조건문에 해당한다. 상위 이미지에서 중간에 있는 빨간색 원 모양이 노선 변경 스위치이다. 노선 변경 스위치는 스위치가 왼쪽을 가리키고 있다면 기차가 왼쪽으로 이동 하고 스위치가 오른쪽을 가리키고 있다면 기차는 오른쪽으로 이동한다. 스위치 선택을 통해서 특정 노선으로 이동하게 하거나 이동하지 못하게 할 수 있다. 왼쪽 이미지에서 T>3 같은 식을 조건식이라 한다. 이전에 배운 조건문은 조건식이 true 인지 false 인지에 따라 진행 방향이 달라 졌는데 이것은 기차 스위치와 비슷하다. 스위치는 기차가 오른 쪽으로 갈 것인지 왼쪽으로 갈 것인지 결정할 수있다. 프로그램에서 조건식은 true이면 왼쪽 false이면 오른쪽 이런 식으로 위에서 아래로만 진행하던 프로그램을 특정 부분만 실행 하거나 실행 하지 않도록 할 수 있다. 프로그램에서 실제로 조건문을 if 문을 사용하여 표현한다. if라는 용어를 기억해 두자. 스위치 부분은 01 부분에 해당한다.

**3번 반복문**

반복문에서 중간의 빨간 원은 노선 변경 스위치이고 2번 조건문과의 차이는 조건문은 왼쪽으로 갈 것인지 오른쪽으로 갈 것인지 결정 할때 스위치를 사용하지만 3번 반복문 같은 경우에는 스위치를 기준으로 오른쪽 원 부분을 반복할 것인지 않을 것 인지를 결정한다. 원은 상황에 따라서 여러번 돌 수도 있지만 한번 또는 돌지 않을 수도 있다. 아래 이미지에서 게임을 진행할 때 말이 01에서 02, 02에서 03, 03에서 04, 04에서 01로 특정 조건에 맞는 동안 반복하는 것을 확인할 수 있다. 이 부분이 반복문에 해당한다. 스위치 부분은 05왼쪽 T==1이라고 쓰여져 있는 부분이 된다. 반복문은 조건문과 마찬가지로 반복 여부를 boolean값으로 결정한다. 결정된 값이 true면 특정 부분을 반복하고 false 이면 반복문을 빠져 나가게 된다는 이야기이다. 프로그램에서 반복문은 for 또는 while 문을 사용하여 표현 한다. 



그림을 보고 검정색 라인이 철도 길이라고 생각 하고 기차가 각각의 철도 길에서 위에서 아래로 이동 했을 경우 어떤 숫자를 지나왔는지 확인해 보자. 프로그래머들은 결과 같을 화면에 보여줄때 출력 한다고 이야기한다. 기차가 지나온 순서대로 화면에 출력해 보자. 빨간원 부분 스위치는 상황에 따라 왼쪽이나 오른쪽 둘중에 하나를 선택해서 이동한다.

그림을 보고 만약 사각형 안의 숫자가 화면에 출력 된다면 윗 부분 01에서 실행이고 아래 부분 끝까지 이동하며 종료라 생각 할 때 각각 화면에 출력되는 숫자를 기술 해보자.

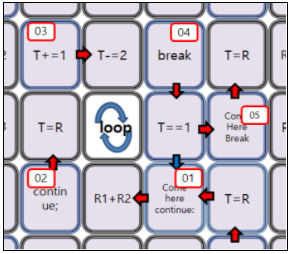
1번의 경우 01,02,03이 출력 되고 다르게 출력이 되는 경우는 없을 것이다. 그래서, 항상 순서대로 실행된다 하여 순차문이라 한다.

2번의 경우 빨간점의 스위치에 따라서 01,02,04 혹은 01,03,04 가 출력 될 것이다. 상황에 따라 2가지 경우중 하나가 선택이 되어 조건문이라고 한다. 프로그램에서 조건문을 사용할때 if문이나 switch문으로 사용된다.

3번의 경우 빨간점의 스위치에 따라 01,03 혹은 01,02,03 혹은 01,02,02,03 혹은 01, 02, 02, .. 02, 03과 같이 02 부분이 여러번 반복될수 있는 결과를 얻게 될것이다. 02 부분이 반복되고 있어서 이부분을 반복문이라고 한다. 반복문의 경우 반복 하고 있는 도중 스위치를 변경하지 않으면 02부분을 무한히 반복하게 될 것인데 이럴 경우를 무한루프 라고 한다. 무한루프가 발생하지 않도록 02 반복문에서 스위치를 변경하는 프로그램을 구현 하여야 한다. 반복문을 프로그램에서 사용 할때 for문이나 while문으로 사용 한다.

다음 문제를 풀어 보자. 02 부분이 안녕을 출력하는 부분 이라면 1,2,3번의 경우 몇 번씩 출력되는지 기술해 보자.

02부분이 화면에 안녕이 출력되는 부분이라고 생각해 보면 1번의 경우에 반드시 한 번만 출력이 되고 2번의 경우 상황에 따라서 한번도 출력 되지 않을 수 있고 한 번만 출력 될 수도 있다. 3번의 경우 상황에 따라서 한번 더 출력되지 않을 수도 있고, 한번 출력 될 수도 있고, 2번 출력 될 수도 있고, 3번 출력 될 수도 있고 여러번 출력 될 수도 있다.

2번 조건문이나 3번 반복문에서 동그라미 원모양 스위치는 T>3와 같은 비교 연산자가 사용된 부분을 조건식 이라고 이야기하고 해당 조건식 실행 결과에 따라 true인지 false 인지에 따라 진행 방향이 결정된다. 왼쪽 이미지에서 반복문을 확인해 보자. 3번 반복문에서 반복 실행 될지 실행 되지 않을지 결정하는 부분은 01번 위쪽의 T==1라는 조건식이 기술된 칸 부분이다. true,false를 결정 짓는 T값을 포함한 조건식을 ‘비교식’이라 이야기 한다. 조건식에 들어오기전 T값을 가지고 있어야만 01에서 비교가 가능하다. 01번 오른쪽 칸에 있는 T=R과 같은 01위치로 들어 오기 바로 전의 T값을 결정하는 식을 ‘초기식’이라고 한다. 다음을 생각 해보자. T값이 초기화 되어서 01번 위치의 비교 값으로 사용되어 반복문을 돌고 있다고 생각해 보자. T 값이 변화 되지 않는다면 반복문을 빠져 나올 수 없다. 이처럼 반복문을 빠져 나오지 못하고 무한히 반복되는 것을 무한 루프라 한다. 반복문을 빠져 나가려면 어떻게 할 것인지 고민해 보자. 반복문을 빠져 나가려면 T값이 변화 되어야 한다. 03 T+=1과 같은 T값이 변화되는 식을 ‘변환식’이라 이야기한다. 반복문이 무한 루프를 돌지 않고 정상적으로 동작 하려면 초기값, 비교값, 변환값이 정상적으로 있어야 한다. 프로그램언어에서 반복문으로 사용되는 While과 for문 에서도 반드시 초기값, 비교값, 변환값이 있어야 한다.

지금 까지 설명한 3가지 문법을 조합하여 컴퓨터에서 돌아가는 모든 프로그램을 만들 수 있다. 프로그램을 만들다가 문제가 발생하면 알고리즘을 사용하여 문제를 해결할 수 있다. 알고리즘에 대해서 알아보자.

알고리즘이란? 어떤 문제를 해결하는 방법을 알고리즘 이라고 한다. 예) 라면을 먹어야 한다면, 슈퍼에가서 라면을 사서 집 부엌에 물을 올리고 라면을 넣어 끓인후 다익으면 먹는다. 여기서 ‘라면을 먹어야 한다면’이 해결 해야 할 문제라면 라면을 먹기 까지의 과정들이 알고리즘에 해당 한다.

프로그램 알고리즘이란? 각종 데이터를 이용해서 사용자가 원하는 결과를 얻어낼 수 있는 프로그램을 구현하는 것을 말한다.

컴퓨터 프로그램이란? 현실 세계의 데이터 정보를 컴퓨터에 넣고 가공 처리 하여 사용자가 원할 때 출력하는 것을 의미 한다. 본인이 물건을 구매하는 웹사이트를 생각해 보자. 사용자로 부터 물건 정보를 입력 받아 저장시켜 놓은 다음에 다른 사용자가 원하는 형태로 저장되어 있는 상품을 보여준다. 대부분의 프로그램의 목적이 이와 같은 정보 처리이다. 여기서 중요한 것이 정보와 처리 부분이다. 3 + 5 라는 계산에서 3, + , 5 각각은 정보에 해당하고 계산한 결과 8은 처리된 결과에 해당 한다.

정보는 컴퓨터에서 다룰때 컴퓨터 자료형 에 넣어 사용하고 데이터 양이 많아지면 데이터 베이스로 관리 되는데 이책에서 다루고 있지 않다. 나중에 JDBC나 database 책에서 다룰 예정이다.

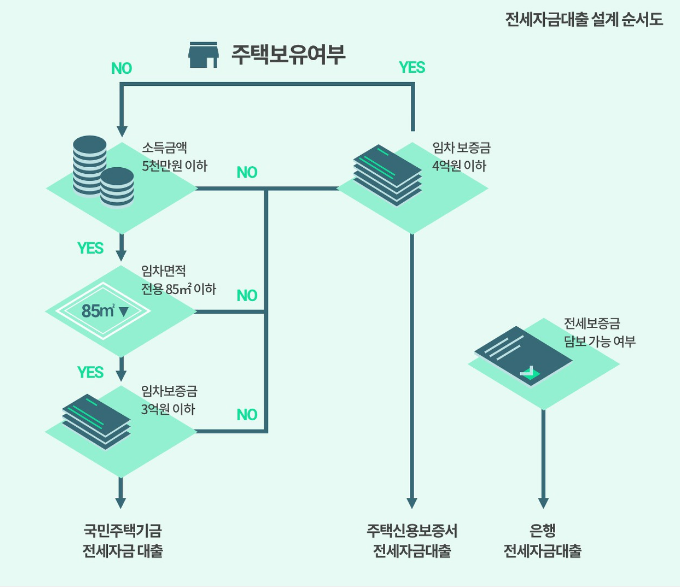
처리는 프로그램 언어로 사용자가 원하는 데로 결과를 보여주는 작업을 해준다. 이때 원하는 결과 대로 보여 주기 위해서는 다양한 방법이 존재 하는데 이때, 어떤 방법을 사용 하여 처리하는 것이 좋을지 고민하게 될 것이고 이 문제를 해결하는 과정을 알고리즘이라고 한다. 알고리즘이란? 문제를 해결하는 방법이다.

컴퓨터 프로그램은 크게 3가지로 구성되어 있다. 순차문, 조건문, 반복문 순차문는 코드가 기술된 순서대로 진행 한다는 이야기이고 조건문는 둘중에 하나을 선택해서 실행 될 수도 되지 않을 수도 있다는 이야기이고, 반복문는 같은 작업을 반복하는 구조를 의미한다. 놀랍게도 상위 3가지 방법을 사용하여 컴퓨터에서 발생하는 모든 문제를 해결 할 수 있다. 결론은 컴퓨터 알고리즘은 순차문, 조건문, 반복문 3가지로 컴퓨터에서 발생한 모든 문제를 해결 할 수 있다.

|  |
| --- |
| > 01. 순서도 사용법 |

컴퓨터에서 알고리즘을 직접 코드로 작성해서 구현 할 수 있지만 알고리즘을 만드는 방법은 쉬운것도 있지만 상당히 어려운 것도 많이 있다. 복잡한 알고리즘을 구현할 때 단순히 코드로 작성하기 보다는 그림 형태의 순서도 영어로 flow chart로 만드는 것이 중요 하다.

순서도란? 어떤 결과를 얻기 위한 과정(알고리즘)을 그림으로 표현한 것이다. 여기에서는 프로그램 알고리즘을 그림으로 표현하는데 사용 되지만 우리 주변에서도 어떤 문제를 해결 하기 위해서 쉽게 만들어 사용하고 있다. 영어로 flow chart라고 한다. 프로그램뿐만 아니라 우리 실생활에서 어떤 문제를 해결하기 위한 과정 풀이로 쉽게 찾아 볼 수 있다. 영화표 구매방법, 세금 혜택 받는 방법, 도서관에서 책대여 하는 방법, 카드 발급받는 방법, 다이어트 하는 방법 등등 순서대로 따라 가다 보면 영화표를 구매할수 있게 되고 세금 해택 대상자인지 아닌지 얼마나 받을 수 있는지, 혹은 원하는 책을 빌릴수 있게 된다.



순서도란? 문제해결을 위해서 해결 방안을 그림으로 표현한 것이다. 생각보다 쉽게 우리 실생활에서 순서도를 찾아 볼수 있다. 어떤 사이트 가입하는 방법, 물건사는 방법, 자격증 취득방법등 컴퓨터 프로그램 문제 해결뿐만 아니라 현실세계의 문제 해결 방법으로 순서도를 많이 사용한다. 상위 순서도 이미지에서 본인이 어떤 종류의 주택담보를 받을 수있는지 확인해보자.

상의 순서도를 이용하여 본인이 내일 배움카드 대상자인지 아닌지 확인해 보자.

알게 모르게 우리 주위에 많은 알고리즘 문제들을 순서도로 그려서 일반사용자들이 사용 및 이해하기 쉽게 해놓은 부분들이 많이 있는데 한번 찾아보자.

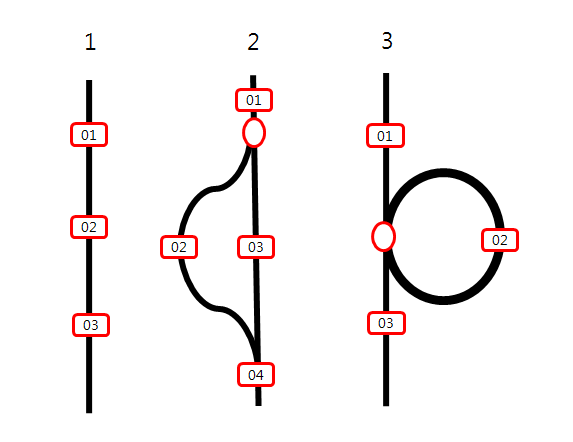
컴퓨터 프로그램관련 자격증 중에 정보처리기사가 있다. 컴퓨터 프로그램 관련 일을 하고 싶다면 정보처리기사를 반드시 취득 하도록 하자. 다른 잡다한 자격증은 필요 없으니 특별한 이유가 없다면 취득하지 않는것이 좋다.

복잡한 알고리즘을 순서도를 이용하여 그림으로 표현한다고 이야기하였다. 다음은 순서도에서 어떤 기호를 사용하여 그리는지 다음 표를 확인해 보자.

|  |  |  |
| --- | --- | --- |
| **기 호** | **이름** | **의 미** |
|  | 시작,  끝 | 순서도의 시작과 끝을 나타낸다. 보통 원안에 시작이나 끝이라고 기술해서 만든다. |
|  | 처리 | 각종 연산, 데이터 처리, 변수의 값변경 등의 작업을 하는데 사용한다. 박스안에 처리 내용을 기술한다. 기차가 01를 지나간다.기차가 02를 지나간다. |
|  | 판단 | 조건 비교 판단하여 상황에 따라 로직의 흐름을 변경함 보통 true 나 false가 리턴될 수 있는 비교 연산자가 기술됨 |
|  | 입출력 | 사용자의 데이터의 입력과 출력이 필요할때 사용한다. 보통 처리에 상황을 기술하여 기술한 내용으로 대처 해서 사용한다. |
|  | 준비 | 데이터 저장 공간을 확보하기 위해 준비 할때 사용한다. 보통 처리로 대처 해서 사용한다. |
|  | 서류 | 종이 같은 실제 출력 최종결과물을 표현한다.  보통 처리로 대처 해서 사용한다. |
|  | 흐름선 | 순서도의 이동 방향을 의미한다. |
|  | 반복 | 사각형 안에 위부분은 몇번 반복 할 수 있는지 조건을 넣고 아래부분에는 반복할 순서도를 넣는다. 왼쪽은 상자안의 순서도를 i=0부터 10보다 작은동안 10변 반복하라는 이야기이다. |

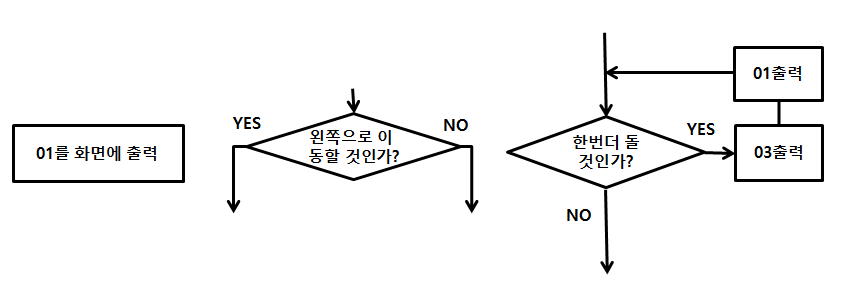
동그라미 모양인 시작과, 끝은 프로그램 시작과 끝에 사용되어 프로그램의 시작과 종료를 의미한다. 하나의 프로그램에는 하나의 시작과 끝이 있도록 구현하여야 한다. 다이아몬드 모양인 판단은 상황에 따라 실행되는 방법이 달라질 때 사용한다. 조건문 if에 해당한다. 직사각형 모양인 처리는 발생하는 작업을 기술할 때 사용한다. 입출력, 준비, 서류 같은 다양한 경우가 있는데 보통 직사각형 모양인 처리로 대체해서 사용 한다. 이 책에서도 입출력, 준비, 서류 같은 경우 정해진 여러가지 도형으로 하지 않고 직사각형 모양에 처리 내용을 기술하여 표현 하였다.

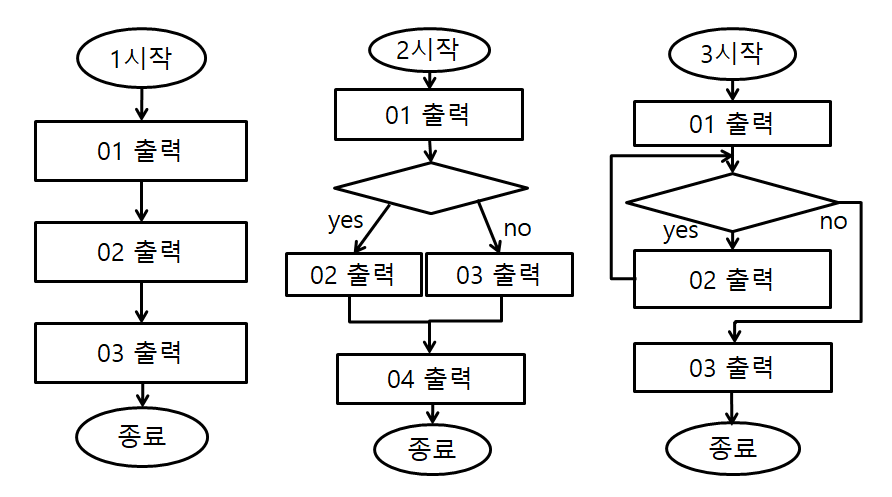
컴퓨터 알고리즘은 순차문, 조건문, 반복문 3가지로 컴퓨터에서 발생하는 모든 문제를 해결 할 수 있다고 하였다. 아래 기차 길에서 기차가 위에서 아래 쪽으로 이동한다고 할때 기차가 이동할 때 지나가는 숫자들이 화면에 출력된다고 생각한다면 1순차문, 2조건문, 3반복문 각각에 대해서 출력 가능한 숫자들을 모두 기술해 보자.

1번 순차문의 경우 01,02,03이 출력 되고 다르게 출력이 되는 경우는 없을 것이다. 2번 조건문의 경우 빨간점의 스위치에 따라서 01,02,04 혹은 01,03,04 가 출력 될 것이다. 3번 반복문의 경우 빨간점의 스위치에 따라 01,03 혹은 01,02,03 혹은 01,02,02,03 혹은 01, 02, 02, .. 02, 03과 같이 02 부분이 여러번 반복될수 있는 결과를 얻게 될것이다. 2번 조건문과 3번 반복문은 비슷한 부분이 있는데 2번의 경우 반복되지 않고 선택 후 종료 되지만 3번의 경우는 상황에 따라 반복 되도록 순서도가 그려진다는 것에 주의하자. 상위 기차길 이미지를 순서도로 변경하면 아래와 같은 모양이 된다.

이미지에 첫번째 순서도가 순차문 두 번째 순서도가 조건문 if, 세 번째 순서도가 반복문 for, while 에 해당한다. if, for, while 문은 프로그램에서 사용하는 약속된 문자 기호 즉, 키워드로 생각 하면된다.

아래 순서도를 확인해서 상위 기차길과 똑같이 화면에 출력 되도록 3개의 순서도를 그려보자. 아래 이미지는 기찻길을 순서도로 그린것이다. 맞게 그렸는지 확인해보자.

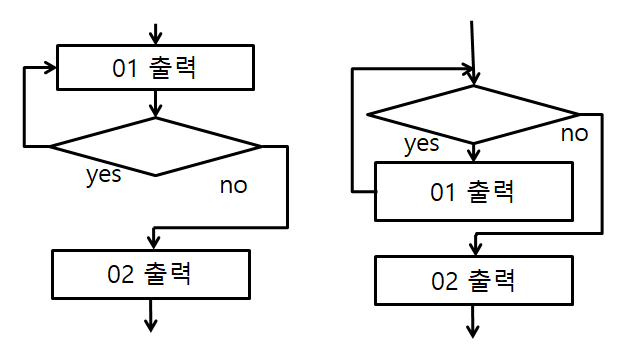


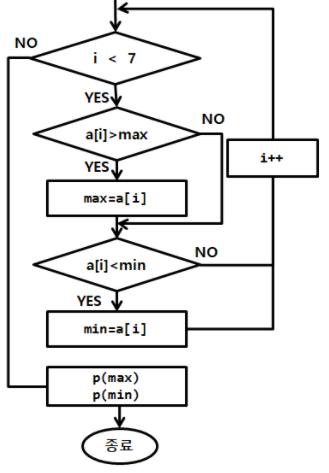


조건문과 반복문은 비슷한 부분이 많다 좀 더 자세히 구분해보면

조건문은 true false로 갈라진 길이 둘다 아래부분 즉 종료쪽으로 이동하다가 언젠가는 한 곳에서 모이게 되는 구조로 되어 있다. 상위 조건문 두번째 순서도에서는 다이아몬드 모양의 조건문에서 갈라졌다가 04 출력 부분에서 합쳐졌다.

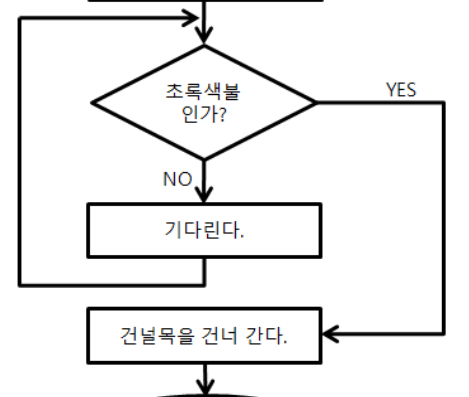
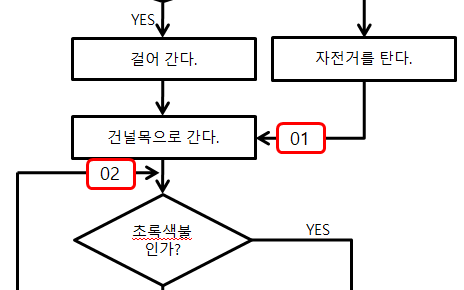
반복문은 한쪽은 계속 조건문으로 들어와서 반복되고 다른 쪽은 반복 문을 빠져 나가게 된다. 상위에서는 다이아몬드 모양의 조건문에서 반복되는 문구는 true 일때 반복 문구 ‘02출력’를 실행한 후 다시 조건문으로 이동하고 false 일 때는 반복문을 빠져나가 ‘03출력’부분으로 이동한다. 실제 순서도를 만들면 복잡한 상태가 되어서 조건문과 반복문을 구분하기 어려울 때가 많다. 천천히 살펴보면 조건문은 treu, false로 선택된 어느쪽으로 이동하더라도 언젠간 만나게 되고 반복문은 어느 한쪽이 조건식 위쪽으로 이동하는 것을 확인할 수 있다. 조건문은 둘 다 조건문 아래쪽으로 이동하지만 반복문은 둘 중 하나는 조건문 은 다시 조건문으로 반복적으로 이동한다.

순서도 그릴때 주의해야 하는 부분을 확인해 보자. 반복문을 순서도로 그릴 때 아래쪽 이미지의 두 순서도 처럼 그리는 일이 있는데 초보자들은 두 순서도를 같은 결과가 나올 것으로 생각하는 경우가 있는데 실질적으로는 다르다. 왼쪽의 경우 01이 반드시 1번 이상 출력되지만 오른쪽 순서도는 한번도 출력되지 않는 결과를 얻을 수 있다. 따라서 전혀 다른 순서도이고 프로그램으로 구현 할 때 반복문은 보통 while이나 for문을 사용 하는데 왼쪽의 경우 do~while문을 사용 해야 하는 경우이다. 반드시 1번 꼭 출력되어야 할 경우는 왼쪽 순서도를 그렇지 않을 경우에는 오른쪽 순서도를 사용하여야 한다. do~while은 1~N번까지 반복할때 while은 0~N번까지 반복할 때 사용하는 것이다.665 do~while를 while로 변경하는 방법 더 낳은 방법은 다음에

옆에 복잡한 순서도가 있는데 여기에 조건문과 반복문이 있다 다이아몬드가 세 개인 걸로 봐서 총 세 개가 있다. 다이아몬드 부분이 조건 문인지 반복 문인지 확인해 보자. 

첫 번째 다이아몬드는 Yes 쪽을 따라가다 보면 결국에는 조건식 다이아몬드로 다시 돌아오는 걸 확인할 수 있다. No 적으로 따라가보면 결국 프로그램이 종료되는 것을 확인할 수 있다. 따라서 첫 번째 다이아몬드는 반복문이다.

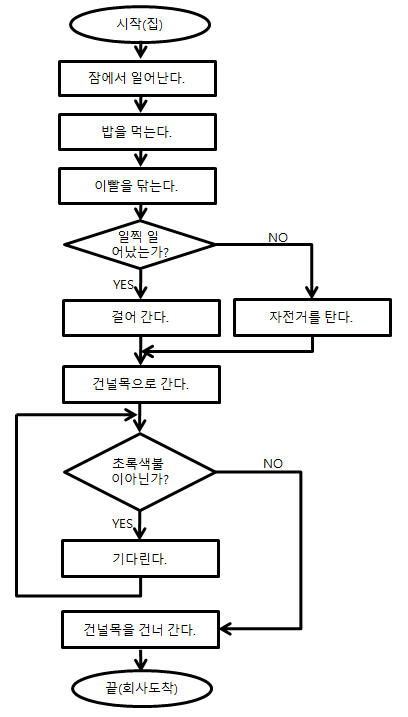
두 번째 다이아몬드 와 세 번째 다이아몬드는 어느 쪽으로 이동 하든 결국에 한 곳에서 만난다.



왼쪽 이미지 처럼 while문 형태를 순서도로 작성할때 종종 왼쪽처럼 false부분이 반복되게 만드는 경우가 있는데 이는 잘못된 순서도는 아니지만 프로그램의 while문은 true가 반복되게 되어 있어서 true부분을 반복되게 만들어야 프로그램 언어로 변경하기 쉽다. 따라서, 반복문으로 만들때 반드시 true부분이 반복되게 만들어야 한다.

오른쪽 이미지에서 01번처럼 순서도 도형 이미지에 직접 흐름을 나타내는 화살표가 들어가는 경우가 많다. 잘못 되었다고 말하기는 어려우나 순서도가 복잡해 지면 순서도를 알아보기 힘들게 될 수 있으니 순서도 도형에 직접들어가는 화살표는 한개만 있는 것이 좋다. 02부분처럼 순서도 도형에 직접 연결 하지 말고 흐름 선에 연결하자.

다음 이미지 순서도는 집에서 회사까지 가는 과정을 순서도로 만든 것 이고 이미지 오른쪽에 있는 코드는 의사 코드이다.

의사코드

시작(집)

잠에서 일어난다.

밥을 먹는다.

이빨을 닦는다.

if(일찍 일어났는가?){

걸어간다.

}else{

자전거를 탄다.

}

건널목으로 간다.

while(초록색불이 아닌가?){

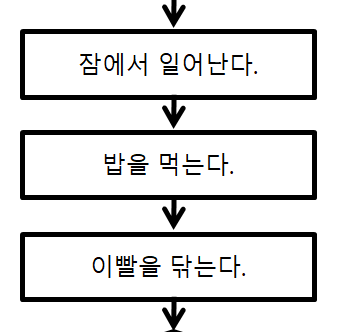
기다린다.

}

건널목을 건너 간다.

회사도착

의사 코드란? 프로그램은 아니지만 프로그램언어처럼 알아보기 쉽게 기술한 코드를 의미한다. 자바에 가까운 형태로 순차문은 그냥 순서대로 글로 기술하였고 조건문은 if문을 반복문은 while문으로 키워드로 사용하여 만들어 보았다. 의사 코드는 순서도를 글로 옮겨 놓은 것이라 생각해도 무관하다. 특정한 문법은 없고 의사 코드라는 의미는 사람이 알아보기 쉽게 기술 하였다고 생각하면 된다.

상위 순서도에서 순차문, 조건문, 반복문을 찾아보자.

왼쪽은 조건문에 해당한다. 다음은 의사코드 이다.

잠에서 일어난다

밥을 먹는다

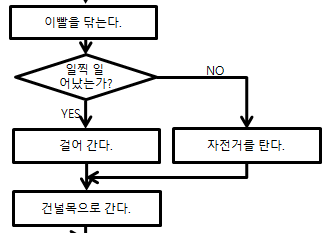
이빨을 닦는다

좀더 실제 프로그램 코드와 유사한 형태로 의사코드 작성을 해 보겠다.

p()의 의미는 소괄호 안의 내용을 화면에 출력 하라는 의미로 생각 하면되고. 상위 일어나고 있는 작업들을 화면에 출력하는 프로그램을 구현하면 다음과 같다.

p(잠에서 일어난다) p(밥을 먹는다) p(이빨을 닦는다)

자바 프로그램 에서는 System.out.println(“잠에서 일어난다”)라고 기술하면 화면에 “잠에서 일어난다” 문자열이 출력되는데 프로그램 언어마다 출력 방법이 조금씩 다르다. 여기 의사코드에서는 서는 줄여서 p(잠에서 일어난다)로 기술 하면 화면에 소괄호 안의 내용이 화면에 출력된다는 의미이다.

다음 부분은 조건문에 해당하는 부분이다.

이빨을 닦는다.

if(일찍 일어났는가?){

걸어간다.

}else{

자전거를 탄다.

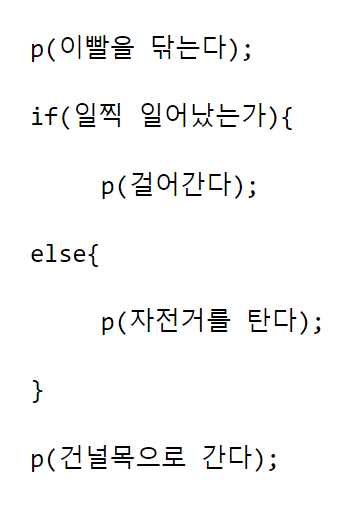
}

건널 먹으러 간다.

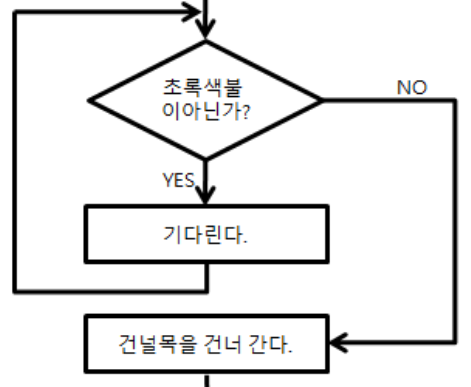
조건문은 참 거짓 실행 부분을 결정한다. 프로그램 언어에서는 조건문을 if문으로 사용 한다. if문 사용방법은 다음과 같다.

|  |
| --- |
| if(조건식){  //참 일때 실행되는 코드 블록  }else{  //거짓일때 실행되는 코드 블록  } |

|  |
| --- |
| if(true) if(a>1) //조건에는 true,false의 결과를 가지는 수식만 올 수 있다 |

if문 다음에는 조건식을 포함한 소괄호가 오고 소괄호 안에 조건식이 참인지 거짓 인지에 따라 실행 되는 부분이 달라진다. 참일 경우에 두 개의 중괄호중에 else 키워드 이전 부분인 첫번째 중괄호 부분이 실행되고 거짓일 때는 else 다음에 오는 두번째 중괄호 블록 부분이 실행된다. 실행할 거짓부분이 없다면 else{} 부분을 생략할 수 있다.왼쪽 이미지는 프로그램 언어로 기술한 결과물이다. 왼쪽 의사코드를 보고 순서도를 그려보자.

아래 이미지 부분은 반복문에 해당한다.

의사코드

while(초록색 불인가?){

기다린다.

}

건널목을 건너 간다.

프로그램 언어에서 반복문은 while과 for 문을 사용 하는데 for문은 축약된 의미가 많아서 순서도를 그대로 옮기기에는 while문에 비해서 어려워 for문 대신에 while문 을 사용하였다. 사용방법은 다음과 같다.

|  |
| --- |
| while(조건식){  //반복할 코드블록  }  // while 문을 빠져 나오면 이어서 실행된 된다 |

while문 사용방법은 while다음에 소괄호로 묶은 조건식을 쓰고 중괄호로 반복할 코드를 기술한다. 조건식이 true면 반복할 코드를 반복하고 false이면 while문을 빠져나간다.

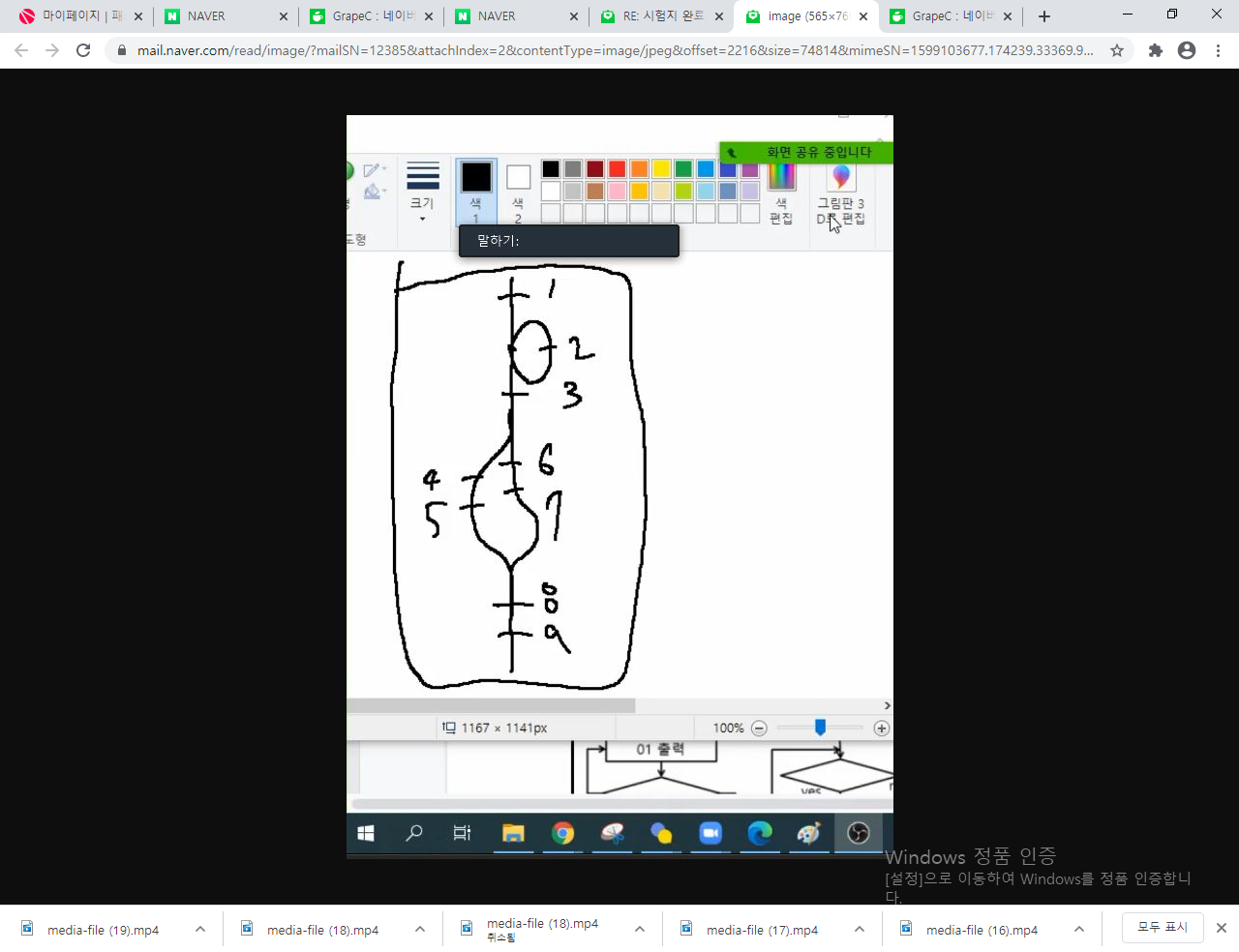
while문의 조건식 부분이 코드블록을 반복할 것인지 말것인지를 결정한다. true이면 반복 할 코드 블록이 실행되고 false이면 while문 중괄호 부분을 빠져나가 실행된다.

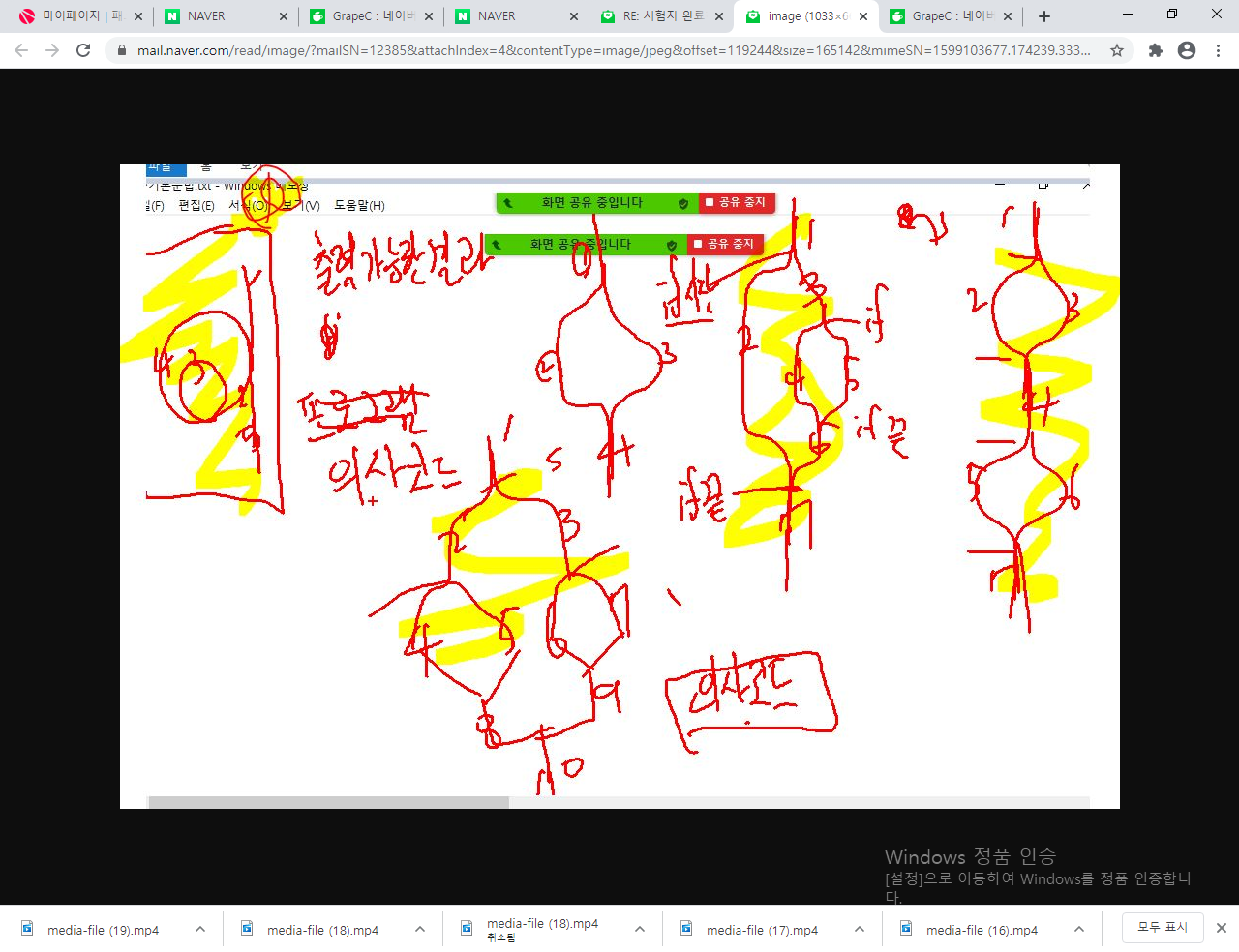
문제1. 우리 주위에서 발생할 수 있는 상황을 순서도와 프로그램 코드로 작성하고 상위 처럼 순차문, 조건문, 반복문 으로 구분하여 작성하여 보자. 상위처럼 전체 순서도,순차문 조건문 반복문 순서도로 총 4개 만들어 보자.

문제2. 라면끓이는 방법을 순서도로 만들어보자.

문제3. 다음을 순서도로 만들어 보자. 버스 대기 시간이 10분이상 넘으면 걸어 가고 이전이면 513번 버스를 타고 다른 번호의 버스는 타지말고 목적지에 가는 순서도를 만들어 보자.

기차길이미지를 문제풀어보자. if문안에 if문





추가 기찻길을 확인해보자.

본인만의 기찻길을 만들고 숫자를 매긴 다음에 의사 코드로 만들어 보자

1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1

\*\*\*\*\*1\*\*\*\*\*1\*\*\*\*\*1\*\*\*\*\*1

21\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1

1\*\*\*\*21\*\*\*\*21\*\*\*\*21\*\*\*\*2

1111\*\*\*\*21111\*\*\*\*21111\*\*\*\*21111\*\*\*\*2

1111\*\*\*\*22221111\*\*\*\*22221111\*\*\*\*2222

111122223333444411112222333344441111222233334444

|  |
| --- |
| > 03. 변수와 우선순위 개념 |

계속해서 변하는 값을 가지는 수을 변수라고 한다. 함수 f(X)=3X+2=Y라고 할때 X가 1이면 Y=5, X가 2이면 Y는 8이 된다. X값에 따라 Y값이 변화 한다. 여기시 X, Y는 변수이고 변하지 않는 3, 2 같은 수를 상수라 한다. 변수와 상수의 의미를 꼭 기억해 보자.

요리를 예로 들어보자. 설탕 소금 마늘 양파 등 다양한 재료가 있다고 생각해 보자. 부엌에 여기 저기 놓여 있으면 재료들이 섞이고 설탕, 소금 같은 비슷한 재료는 알아 볼 수 없게 될 것이다. 여기서 두가지 문제를 찾아 보자. 첫 번째는 재료가 손실 및 섞일 수 있다는 것 두번째는 비슷한 재료는 구분하기 힘들다는 것 이런 문제를 해결 하기 위해서 접시를 사용한다. 접시에 재료를 담아 두고 접시에 라벨을 붙여 두면 두가지 문제가 해결된다.

실제 컴퓨터에서 설탕 소금 같은 것들을 데이터라하고 어런 것들을 담은 접시들은 메모리에 해당한다. 실제 컴퓨터에서 메모리에 저장된 데이터를 섞이지 않고 구분하기 위해서 변수를 사용한다.

상위 두 상황을 합쳐서 생각해 보면 함수 f(X)=3X+2=Y에서 변수 x, y는 요리 할 때 사용하는 소금 접시, 설탕 접시에 해당한다. 변수에 이름을 붙여서 x,y 로 사용하는 것처럼 접시에 이름을 붙여 소금접시, 설탕접시가 된 것이다. X에 3를 넣는다는 의미의 X=3은 소금 접시에 소금 10G를 넣는다는 의미의 ‘소금접시 = 소금 10그램’과 같이 표현한 것이다.

‘소금접시 = 소금10그램, 소금접시=설탕 5그램’ 옆의 2가지 문구를 순서대로 적용 한다면 소금접시에 어떤 결과가 남을 것 인지 생각해 보자. = 은 기존 값에 새로운 값을 할당 하라는 의미이다. 따라서, 접시에는 마지막 값 설탕 5그램이 들어 있다.

초임들이 많이 하는 질문을 살펴보면 다음과 같은 것이 있다. 이전에 접시에 담겨 있는 소금은 10그램은 어디로 갔는가? 사라졌다고 생각하자. 소금 접시에 왜 설탕이 담겨져 있는가? 접시 이름을 적절하게 붙이지 못한 것 이지 잘못된 것은 아니다. 만약에 소금 10 그램의 의미가 중요 하다면 다른 접시에 따로 보관하여야 했다. 결론적으로 ‘소금접시 = 소금 10그램, 소금접시=설탕 5그램’ 에서 ‘접시1 = 소금 10그램, 접시2=설탕 5그램’ 으로 변경한 후 접시1, 접시2를 구분하여 사용하여야 한다.

a=5, a=6을 순서대로 실행 시킨 후 실행이란 컴퓨터가 한줄 한줄 읽고 적용시키는 것을 의미한다. 실행 시킨 a값은 6이 된다. 그럼 과거의 5값은 어떻게 되느냐 사라진다고 생각해도 무방하다. 만약에 과거의 5값을 접근할 수 있는 방법이 필요 했다면 다음과 같이 다른 변수에 저장 했어야 한다. a=5, b=6 이렇게 하면 a를 통해서는 5 b를 통해서는 6 값을 접근할 수 있다. 이렇게 변수에 저장하고 나면 a=5, 이므로 a대신에 5 b 대신에 6를 사용할 수 있다. 따라서 5+6,a+6,a+b,5+b 다 같은 의미이다.

= 연산자는 새로운 값으로 할당하면 기존의 값은 없어 진다.

변수를 사용하는 이유를 다시 한 번 확인해 보자. 변수는 식별과 저장 공간 확보로 사용 된다.

5+6 의 계산 결과를 sum에 넣어 sum를 출력해보자.

sum=5+6;

sum 출력

다음과 같은 과정을 거치면 sum의 값 11이 화면에 출력 될 것 이다. 상위 과정에 이어서 5를 6로 바꾸고 두수를 더해서 sum에 넣어 출력해 보자.

sum=5+6;

sum 출력

sum=6+6;

sum 출력

첫번째 sum 출력은 11이 출력 될것이고 두번째 sum 출력은 12가 출력 될것이다. 이전 코드의 6를 7로 변경한 다음 sum에 넣어 출력해 보자.

음~? 이전 코드의 6이란 맨처음 출현한 6를 의미하는 것인지 아니면 같은 라인에서 앞쪽에 있는 6을 의미하는 것인지 애매하다.

상황이 sum=7+6;이 될수도 있고 sum=6+7이 될수도 있다.

명확함을 중요시하는 컴퓨터 프로그램에서는 치명적인 의사소통 문제가 될수 있어서 변하는 수를 명확하게 식별하기 위해서 변수를 사용한다. 6의 또 다른 이름이라 생각하면 되고 값이 계속 변하니 변수라고 생각하면 된다. 상위 문제를 다음과 같이 바꿔 보겠다.

a에 5를 넣고 b에 6를 넣은 다음 a, b를 더한 수를 sum에 넣고 a=7로 변경한 다음 a, b, sum를 출력하는 방법을 수식으로 만들어 보자. 출력값이 어떻게 되는가 a=5, b=6, sum=a+b ,a=7 , a 출력, b 출력, sum 출력 과 같이 하면 될 것이다. a는 7이 출력 될것이고 b는 6이 출력될 것이고 sum은 11인가 13인가? 프로그램은 위에서 아래로 순서대로 출력되므로 세번 째 sum=a+b 부분에서 의 a,b값은 5,6이니 sum는 11이 된다.

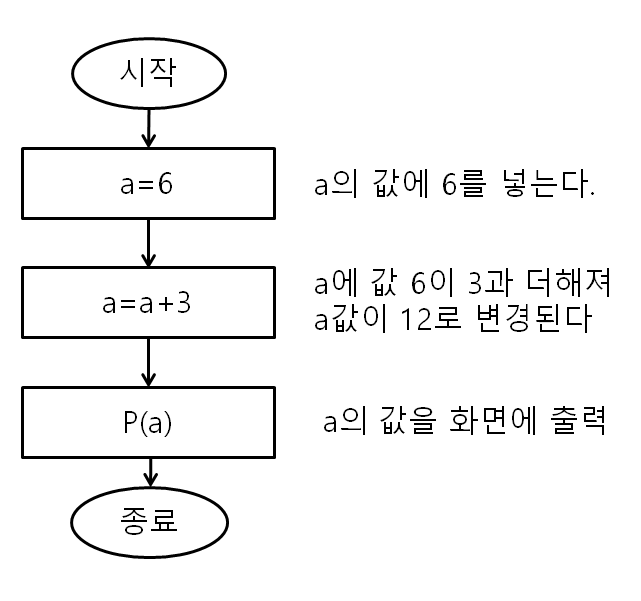
순서도로 상위 글대로 그려보자……. 6666

다음을 또 살펴 보자 설탕접시에 설탕이 5그램 있다고 생각해 보자. 설탕접시에 추가로 10그램을 더 넣어 15그램이 들어있는 설탕 접시를 만들 수 있겠는가? 다음과 같이 하면 된다.

(1)설탕접시 = 5그램, (2)설탕접시 = (3)설탕접시+10그램

(숫자)이 부분은 원래 없는 부분인데 설탕접시가 여러번 나와서 구분하려고 넣었고 (3)번째에 설탕접시는 (1)에서 넣은 5그램의 설탕을 의미하는 거여서 설탕접시 = 5그램 + 10그램과 동일하다. (1)번접시의 5그램의 값은 사라지고 15그램의 값만 남아 있다.

기존 소금접시에 소금10g이 들어 있는데 소금5g를 추가여 접시소금15g이 되도록 하려면 어떻게 해야 하는지 생각해 보자. 소금접시에 10g이 들어 있어야 하므로 ‘소금접시=10g’과 같이 하면되고 소금접시 10g에서 5g을 추가해서 15g으로 만드려면 기존의 소금접시 있는 소금에 추가로 소금 5g를 넣어서 다시 소금접시에 넣어야 하므로 ‘소금접시=소금접시 + 소금 5g’ 이라고 한면 된다.

a값에 6을 넣고 a의 값을 3증가 시킨 다음 a를 화면에 출력하는 순서도를 작성해 보자.

변수 a에 2가 들어 있는데 3를 더하여 a에 값을 5로 변경하려면 다음과 같이 하면 된다. ‘a=2, a=a+3’ 과 같이 하면 된다.

연산자는 우선순위가 있으므로 수식을 만들때 조심하자.

5+4+6 대부분의 연산자는 왼쪽에서 오른쪽으로 연산된다. 5+4가 계산되어 9가 되면 그 다음 남은 연산 9+6이 진행되어 15가된다. 왼쪽에서 순서대로 연산 된다.

a=b=c 처럼 아닌 연산자도 있다 = 연산자의 경우 오른쪽에서 왼쪽으로 할당 한다. c가 1이면 a, b가 1이된다. 왼쪽에서 오른쪽으로 계산된다면 a가 2라면 모두 2가 되겠지만 오른쪽에서 왼쪽으로 계산되므로 처음에 a가 2 c가 1이라면 a,b,c 모두 1이 될것 이다. 따라서, a=1 이 맞지만 1=a 는 잘못된 수식 이다. 상수는 변경할 수 없다. 1을 다른 값으로 변경하는 것은 불가능한 일이다.

5+3\*6, (5+3)\*6 다음 두가지 수식을 연산하면 각각 23,48 이다. +,-,%,/,\* 연산자에서 %,/,\* 가 먼저 계산되어 나온 결과이다. 소괄호 안의 연산이 먼저 실행 된다. 연산자에 따라 순서대로 계산 되지 않고 같은 수식에서 먼저 계산되는 것들이 있다.\*,/,%가 대표적이고 %는 나머지를 얻는 계산으로 /,\*와 우선순위가 같다.

정리해 보자면 연산자에서 중요한 2가지는 연산자 우선순위와 연산 방향이다. 계산 결과가 생각한 것과 다르게 나오면 웹에서 연산자 우선순위를 확인해 보는 습관을 기르자.

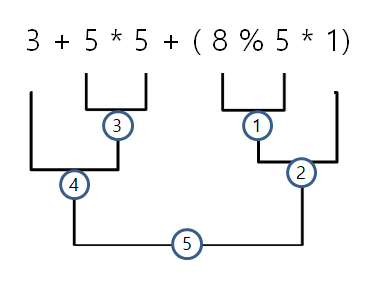
5+3\*6를 한줄에 하나씩 연산하여 연산 과정을 변수 sum에 값을 저장한다면 다음과 같이 된다. sum=0,sum=3\*6,sum=5+sum,p(sum) 마지막에 p(sum)은 sum에 들어있는 값을 출력하라는 이야기이다. P()에 변수가 오면 변수에 들어 있는 값을 그대로 출력하라는 의미이고 변수가 아닌값이 오면 그대로 출력 하라는 이야기이다.

두수식 연산해서 변수에 넣어 순서도로 만들어보자 6666

(5+3)\*6를 각각 한줄에 하나씩 연산해보면 5+3,8\*6,48 이된다.

결과를 순서도로 만들어 보자.

다음 문제를 풀어 보자.



5%6+(3+3\*2)를 한줄에 하나씩 연산해 보자.

=연산자는 +,-,%,/,\* 보다 연산자 우선순위가 낮아서 나중에 실행 된다. a=4+5 오른쪽 부분이 먼저 실행되어서 9가된 다음에 a에 9가 들어 가 실행 결과 a는 9이다. 만약 = 연산자가 우선순위가 높아 먼저 실행되어 a= a에 4가 들어간 다음 a+5가 실행 된다면 실행결과 a는 4가 된다. 하지만 우선순위가 = 이 더 낫기 때문에 a 가 4가 되는 일은 없을 것이다.

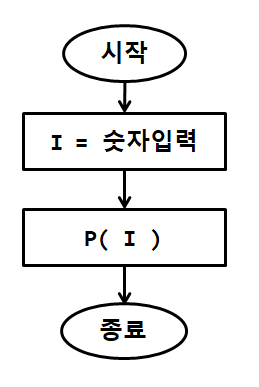
3개 필요한 수식문제를 순서도로 만들어보자.

2%6+3+(2+5)+3/2

상위에서는 우선순위를 공부하려고 2개씩 계산한느것으로 구현하였다. 프로그램에서는 한줄로 구현하여도 상관이없다.

===

변수를 이용하여 사용자 입력을 받아 그때 그때 다르게 출력되는 순서도를 만들어 보자. 변수에 사용자로 부터 숫자 하나를 입력 받고 싶으면 ‘a=숫자입력’ 이라고 기술하면 된다.다음은숫자를 입력 받아 입력한 숫자를 출력하는 순서도와 프로그램이다.

프로그래밍언어

I=숫자입력

P(I)

1. 세로와 가로를 입력 받아 사각형의 넓이를 만드는 순서도와 프로그램을 만들어 보자.

2. 세로 가로 높이를 입력 받아 사각기둥의 부피를 만드는 순서도와 프로그램을 만들어 보자.

4. 입력한 숫자가 10보다 큰 수인지 아닌지 출력하는 순서도와 프로그램을 만들어 보자.

5. 입력 받은 숫자가 양수인지 0인지 음수인지 판단하는 순서도와 프로그램을 만들어 보자.

6. 입력 받은 숫자가 홀수인지 짝수인지 판단하는 순서도와 프로그램을 만들어 보자. 홀수의 정의는 2로 나눈 나머지가 1이면 홀 수 0이면 짝수이다.

7. a센치미터가 몇 미터 몇 센치인지 출력하는 프로그램을 구현하시오.

8. 잔디의 가격은 1 제곱 미터에 a원 일때 위변이 b,아래변이 c, 높이d인 사다리꼴에 잔디를 심으려면 얼마의 비용이 드는지 구하는 순서도를 작성하시오.

%연산의 이해

9. a 원을 500원 짜리 동전으로 바꾸면 500원짜리 몇 개 100원짜리 몇 개가 되는가?

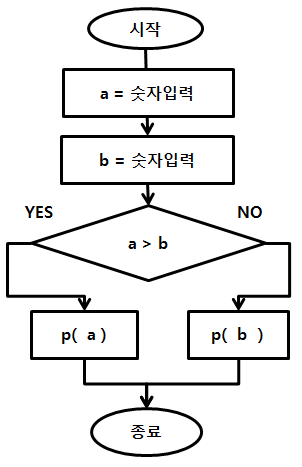
10. 500원짜리 n개 100원짜리 m개 총 얼마인가?

11. n초를 입력 받아 시분초로 바꿔보자.

12. n시간 m분 l초는 총 몇초인가?

교환방법

3. a, b 의 수를 입력 받아 두 수를 교환한 다음에 두 수를 출력하는 순서도와 프로그램을 작성해 보자.

사용자에게 두 수 a, b를 입력 하여 중 큰수를 화면에 출력하는 순서도와 프로그램을 작성해 보자.

프로그램언어

a=숫자입력

b=숫자입력

if(a>b){

p(a)

}else{

p(b)

}

13. A=5,b=6이라면 두수 a,b를 교환해서 a=6,b=5로 만든다음 출력하는 방법을 고민해보자.

힌트. 각 컵의 음료수를 교환하는 프로그램을 만들어 보자.

7. 2개의 수를 a,b에 입력 받아 큰수를 a에 넣은 후 a,b를 출력하여 큰수가 먼저 나오게 해보자.

7. 3개의 수를 입력 받아 작은수부터 출력해보자.

비교연산을 하여 a b c를 교환하여 a b c를 출력 한결과가 아래와 같이 나오도록한다.

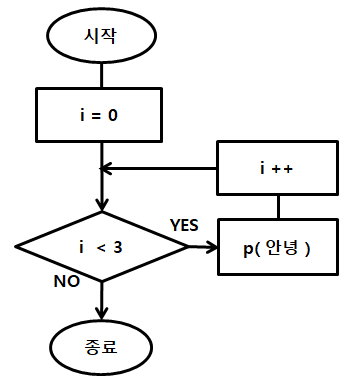
A. 5,1,2이면 1,2,5가 출력

B. 3,7,4이면 3,4,7이 출력

C. 3,5,6이면 3,5,6이 출력

반복문에 대해서 만들어보자.

while 사용법

i=0

while(i<3){

p(안녕)

i++

}

for문 사용법

for(i=0;i<3;i++){

p(안녕)

}

반복문이 참인 동안 중괄호 안의 코드 블록을 반복한다. 상위 순서도에서는 위에 설명한 while문 의사코드만으로도 충분히 구현 가능 하지만 프로그램 언어에서 while문을 사용하려면 아래 처럼 좀더 세분화하여 표현할 필요가 있다.

|  |
| --- |
| 초기값;  while(비교값){  //반복할 코드블록  변환값;  } |

while문은 for문으로 변경할 수 있다.

|  |
| --- |
| for(초기값;비교값;변환값){  //반복할 코드 블록  } |

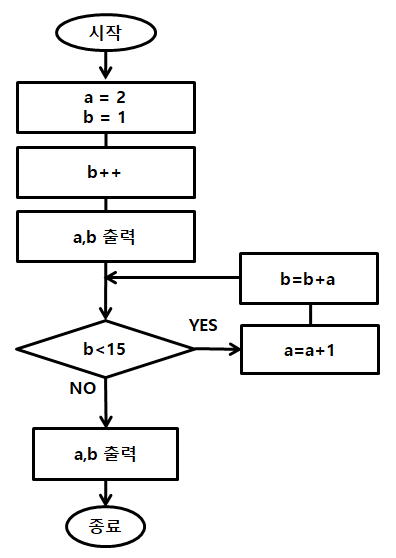
초기값은? 반복할 코드 블록이 실행되기전에 조건식과 관련된 값을 처음 결정하는 부분이다. 초기값은 while문의 반복을 결정하는 비교값과 관련이 있어서 값을 잘못 결정하면 문제가 발생하여 무한루프가 발생 할 수 도 있어서 조심히 사용하여야 한다.

비교값은? 다시 반복할 것인지를 결정할때 초기값과 비교 대상이 되는 값이다. 비교 값이 적절하지 않으면 프로그램이 멈추거나 무한루프가 발생 할 수 있다.

변환값은? 초기값이 변하지 않으면 while문에서 비교값과 비교하였을때 결과가 바뀌지 않아 무한루프를 돌수 있다. 일정 횟수 반복 하다가 while문을 빠져나갈 수 있도록 적절한 변환값을 만들어 주어야한다.

반복문을 잘못 기술하게 되면 반복할 코드블록이 계속 반복하게되는 무한루프가 되거나 한번도 실행하지 않고 반복문을 빠져나가게 된다. 이런 문제를 해결하지 않게 하기 위해서는 반드시 초기값, 비교값, 변환값이 있어야 한다.이후 나올 순서도에 반복문이 있다면 잘못 구현하지 않도록 재대로 구현하여야 한다.

다음 순서도 그림을 확인해 보고 초기값, 비교값, 반환 값에 해당하는 i값을 찾아보고 어떻게 프로그램으로 변환되는지 확인해 보자.

다음 순서도를 프로그램언어로 구현하여 보자. 3식을 찾아보자.

프로그램언어

a=2

b=1

b++

p(a)

p(b)

while(b<15){

a=a+1

b=b+a

}

p(a)

p(b)

프로그램이 종료될 때 출력되는 값들을 기술해보자.

이전 내용들을 참고 하여 다음 문제의 순서도와 프로그램을 만들어 보자.

정답은 맨 아래에 존재 한다.

1을 3번 더하는 순서도

3부터 10을 출력하는 순서도

1~10까지의 사이 숫자를 더한 총합을 sum에 넣어 출력 하기

15~30까지의 사이숫자를 더하기 더한 총합을 sum에 넣어 출력 하기

1~50까지의 짝수를 출력하기

1~100사이의 10의 배수 출력하기

문제5. 주사위 놀이를 하던 판을 순서도와 게임판을 순서도로 그리고 프로그램언어로 완성해 보자.

문제5.우리가 사용하는자판기를 순서도로 자세히 만들어 보자.

두수를 입력 받아 사이에 있는 짝수 구하기.

7. 1 부터 10까지 더하는 순서도와 프로그램을 만들어 보자.

8. 숫자를 하나 입력 받아 1부터 입력한 수까지 순서대로 화면에 출력 되도록 순서도와 프로그램을 만들어 보자.

9. 사용자에게 처음에는 작은수 두 번째는 큰수를 입력 받아 두 수의 사이에 있는 모든 수를 합하는 순서도와 프로그램을 만들어 보자. 예)5 9 입력시 6 7 8 더한 결과를 얻음

14. 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

15. 1/2+2/3+3/4+4/5+ 5/6+6/7+....+99/100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

16. 피보나치 수열을 10개를 순서대로 출력하는 순서도와 프로그램을 작성해 보자. 다음 순서도는 반복해서 사용자가 원하는 메뉴를 입력받아서 보여주는 순서도와 프로그램이다.

10. 원하는 색의 전구와 밝기를 입력 받아 처리하는 순서도와 프로그램을 만들어 보자. 아래 내용들이 들어가 있어야 한다.

초기 변수값 : color=”빨강” brightness =50

값의 범위 : color=빨강,노랑,파랑 brightness는 0~100

제안사항 : brightness의 숫자 변경은 1씩 가능하다. brightness값이 10이라면 다음 brightness값은 11 이나 9 만 가능하다. (반복문 사용)

사용자입력변수 : colorInput, brightnessInput

최종결과값출력 : p(“현재 색상은”+color+”밝기는”+brightness+”이다”);

11. 1~100까지의 7의 배수를 출력하는 순서도와 프로그램을 작성해 보자. 어떤 숫자가 배수 인지 아닌지 알고 싶으면 7로 나눈 나머지가 0이면 7의 배수이고 아니면 7의 배수가 아니다.

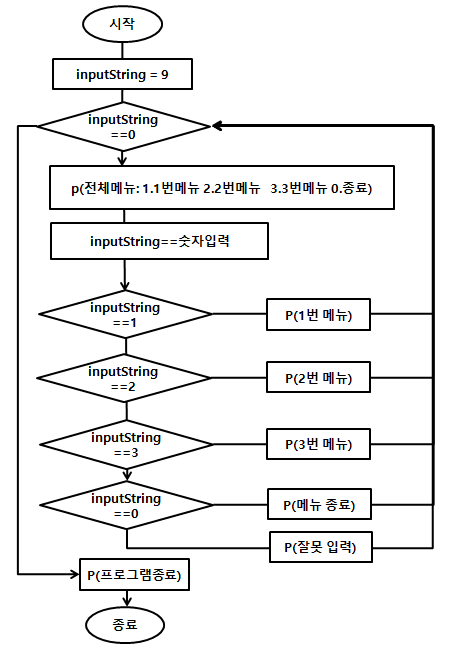
12. 1~100까지의 숫자중 4의 배수를 제외한 모든 수를 출력하는 순서도와 프로그램을 작성해 보자.

13. 1~100까지의 짝수의 합을 출력하는 순서도와 프로그램을 작성해 보자.

14. 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

15. 1/2+2/3+3/4+4/5+ 5/6+6/7+....+99/100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

16. 피보나치 수열을 10개를 순서대로 출력하는 순서도와 프로그램을 작성해 보자. 다음 순서도는 반복해서 사용자가 원하는 메뉴를 입력받아서 보여주는 순서도와 프로그램이다.



다음 왼쪽을 확인해 보자. ‘전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료’ 다음과 같은 문자열이 화면에 나오고 사용자가 숫자를 입력 받아 사용자가 선택한 번호의 메뉴가 화면에 나온다. 1번을 선택하면 ‘1번 메뉴입니다.’ 출력하고 2번을 선택하면 ‘2번 메뉴입니다.’이 출력되고 0를 입력하면 프로그램이 종료 된고 1~3번의 경우는 해당 번호의 메뉴를 출력하고 다시 전체메뉴가 출력되어 사용자 입력을 기다린다.

inputString = 9

while (inputString==0) {

p(전체메뉴: 1.1번메뉴 2.2번메뉴 3.3번메뉴 0.종료)

inputString = 숫자입력

if(inputString==1) {

p("1번 메뉴")

}else {

if(inputString==2) {

p("2번 메뉴")

}else {

if(inputString==3) {

p("3번 메뉴")

}else {

if(inputString==0) {

p("메뉴 종료")

}else {

p("잘못 입력")

}

}

}

}

}

p("프로그램종료");

상위 if문을 줄여서 다음과 같이 else if문으로 바꿀 수 있다. 다음을 확인하자.

if(inputString==1) {

p("1번 메뉴")

}else if(inputString==2) {

p("2번 메뉴")

}else if(inputString==3) {

p("3번 메뉴")

}else if(inputString==0) {

p("메뉴 종료")

}else {

p("잘못 입력")

}

17. 은행 프로그램을 순서도와 프로그래밍 언어로 만들어보자.

은행 프로그램의 메뉴는 다음과 같다. ‘전체메뉴: 1.입금 2.출금 3.조회 0.종료’

account 변수에 처음에 0으로 세팅하고 입금 출금을 통에 원하는 액수를 더하거나 빼준다. 결국 account 변수에 있는 숫자가 은행계좌에 남은 돈이 된다.

입금 메뉴를 통해 입금 액을 받아 account 변수에 추가 할 수 있다.

출금 메뉴를 통해 출금 액을 account 변수에서 뺄 수 있다.

조회 메뉴를 통해 account에 입금 액이 얼마나 남아 있나 확인 할 수 있다.

18. 금액을 입력받아 천원, 오백원, 백원, 오십원, 십원 짜리 잔돈으로 거슬러주는 프로그램과 순서도을 구현해 보자. 예제 금액을 1000으로 나눈 몫은 돈을 거슬러 주었을때 천원짜리 갯수 이고 1000으로 나눈 나머지는 천원짜리로 환산하고 남은 잔돈이다. 10원 이하는 입력받지 않는다. 최종 결과물은 다음과 같이 될 것이다. 5820원을 잔돈으로 바꾸면 천원짜리 5개, 오백원짜리 1개, 백원짜리 3개, 오십원짜리 0개,십원짜리 2개

선언할 변수:change1000,change500,change100,change50,change10,moneyInput

출력값:p(“천원짜리 “+change1000+”개, 오백원짜리 “+change500+”개, 백원짜리”+ change100+”개, 오십원짜리 “+change50+”개, 십원짜리 “+change10개)

19. 실존하는 자판기와 동일하게 만들어 보자.

메뉴:

남은돈 0원.

1.사이다 700 2.콜라 500 3.환타 350 4.투입 5.반환

사용자 입력 변수 :

inputString, moneyInput, moneyTotal

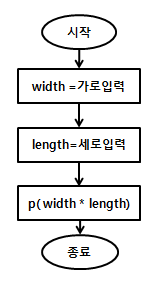
시나리오 :

천원짜리 세장 넣고 사이다1개 콜라2개 환타1개 를 구매한다.

출력값:

사이다 콜라 콜라 환타 천원짜리 0개 오백원짜리 1개 백원짜리 4개 오십원 1개

|  |
| --- |
| > 05.순서도 답안 |



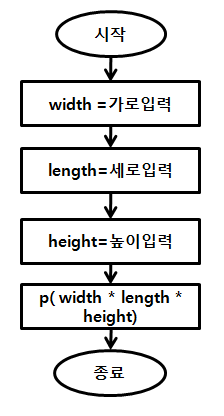
1. 세로와 가로를 입력 받아 사각형의 넓이를 만드는 순서도와 프로그램을 만들어 보자.

프로그램언어

width=가로입력

length=세로입력

p(width\*length)



2. 세로 가로 높이를 입력 받아 사각기둥의 부피를 만드는 순서도와 프로그램을 만들어 보자.

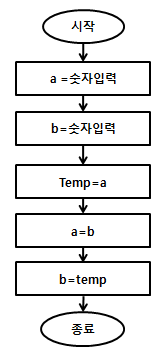
프로그램언어

width=가로입력

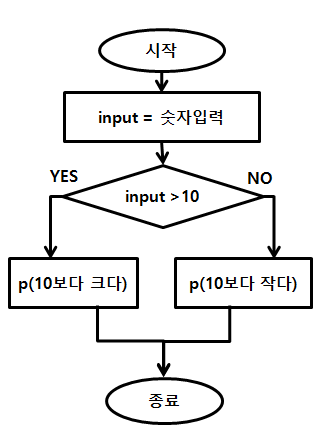
length=세로입력

height=높이입력

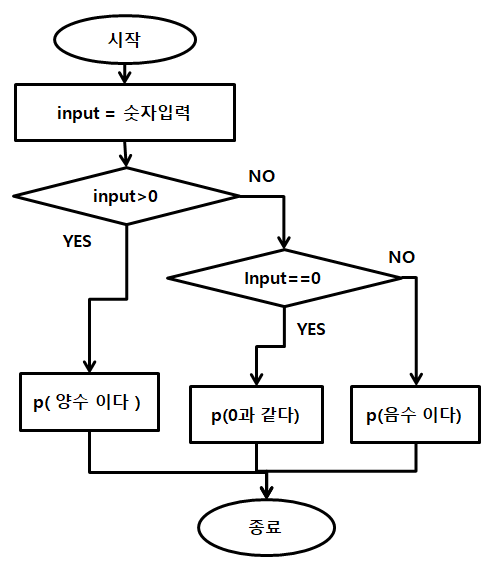
p(width\*length\*height)



3. a, b 의 수를 입력 받아 두 수를 교환한 다음에 두 수를 출력하는 순서도와 프로그램을 작성해 보자.



4. 입력한 숫자가 10보다 큰 수인지 아닌지 출력하는 순서도와 프로그램을 만들어 보자.



5. 입력 받은 숫자가 양수인지 0인지 음수인지 판단하는 순서도와 프로그램을 만들어 보자.

input=숫자입력

if(input>0){

p(양수 이다)

}else{

if(input==0){

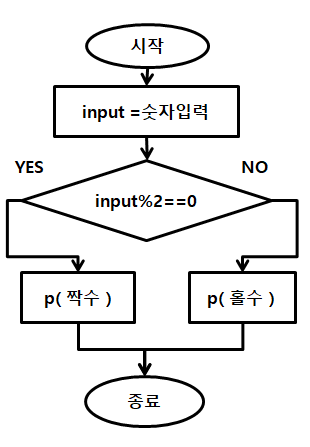
p(0과 같다)

}else{

p(음수 이다)

}

}

6. 입력 받은 숫자가 홀수인지 짝수인지 판단하는 순서도와 프로그램을 만들어 보자. 

input = 숫자입력

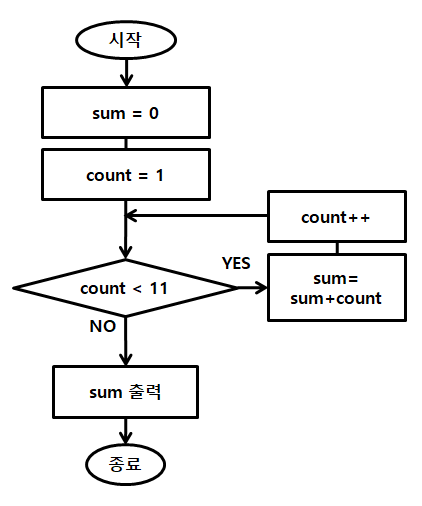
if( input%2==0){

p(짝수)

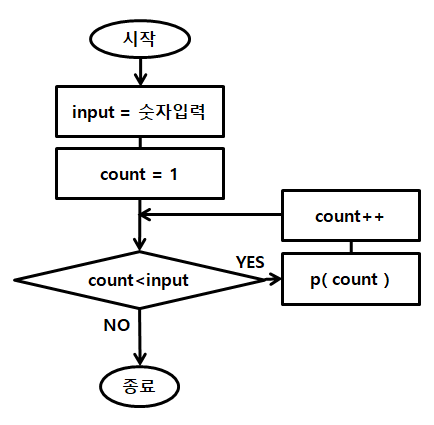
else{

p(홀수)

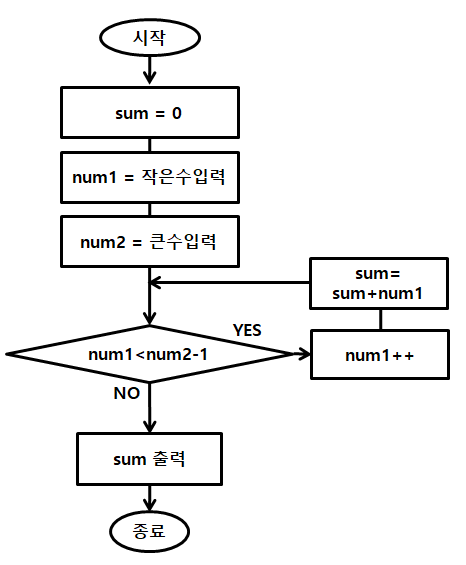
}

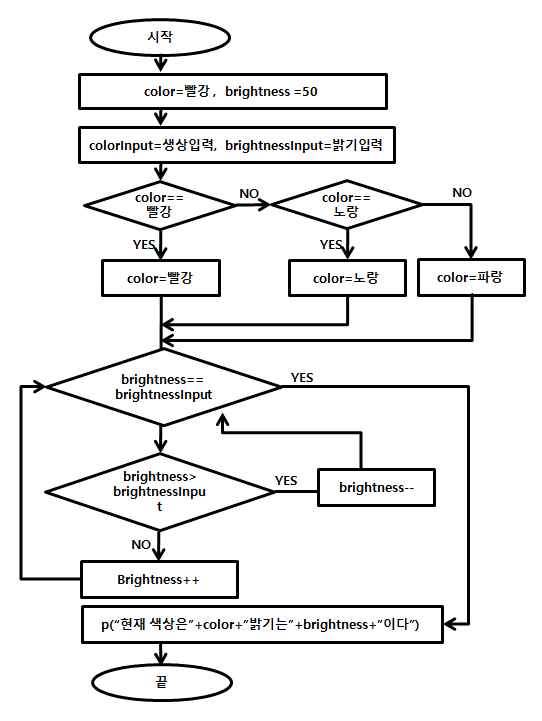


7. 1 부터 10까지 더하는 순서도와 프로그램을 만들어 보자.

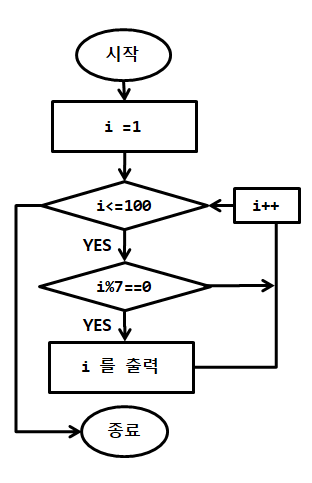


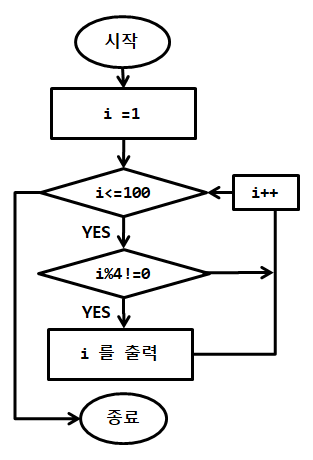
8. 숫자를 하나 입력 받아 1부터 입력한 수까지 순서대로 화면에 출력 되도록 순서도와 프로그램을 만들어 보자.

9. 사용자에게 처음에는 작은수 두 번째는 큰수를 입력 받아 두 수의 사이에 있는 모든 수를 합하는 순서도와 프로그램을 만들어 보자. 

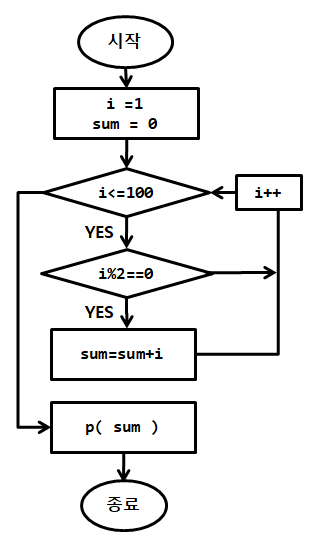


10. 원하는 색의 전구와 밝기를 입력 받아 처리하는 순서도와 프로그램을 만들어 보자. 아래 내용들이 들어가 있어야 한다.

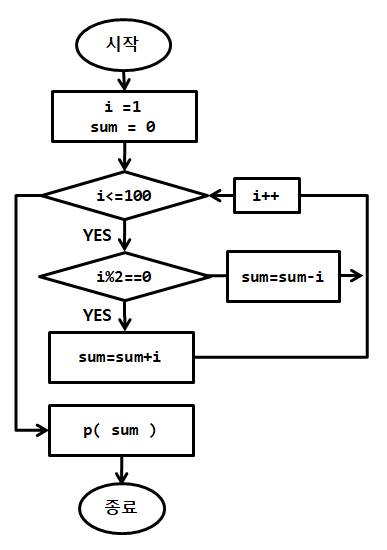
11. 1~100까지의 7의 배수를 출력하는 순서도와 프로그램을 작성해 보자. 



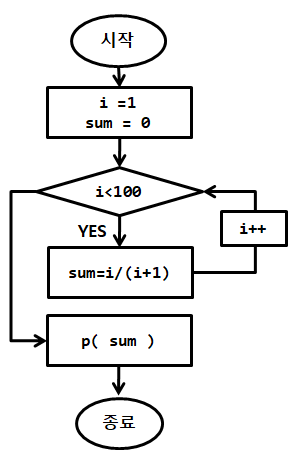
12. 1~100까지의 숫자중 4의 배수를 제외한 모든 수를 출력하는 순서도와 프로그램을 작성해 보자.

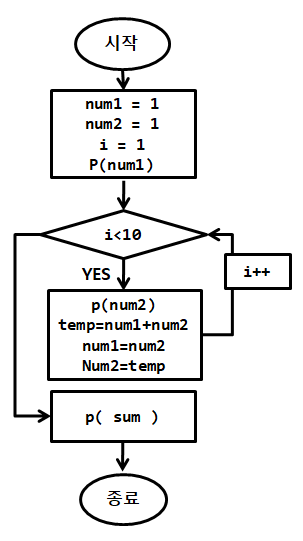


13. 1~100까지의 짝수의 합을 출력하는 순서도와 프로그램을 작성해 보자.

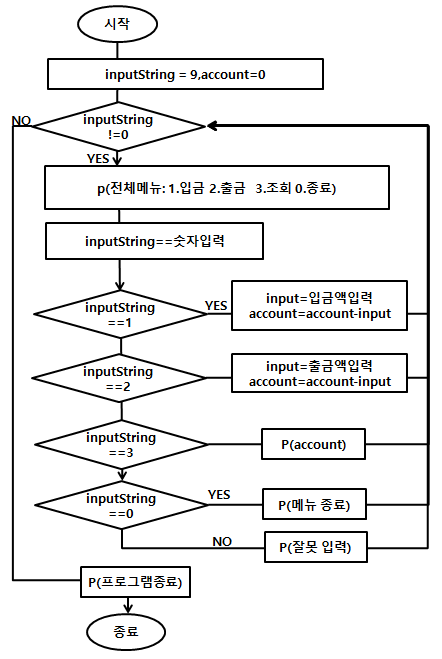


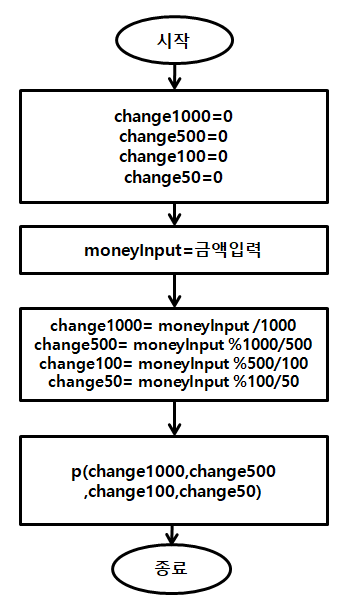
14. 1-2+3-4+5-6……+99-100의 결과를 구하는 순서도와 프로그램을 작성해 보자.

15. 1/2+2/3+3/4+4/5+5/6+6/7+....+99/100 의 결과를 구하는 순서도와 프로그램을 작성해 보자.



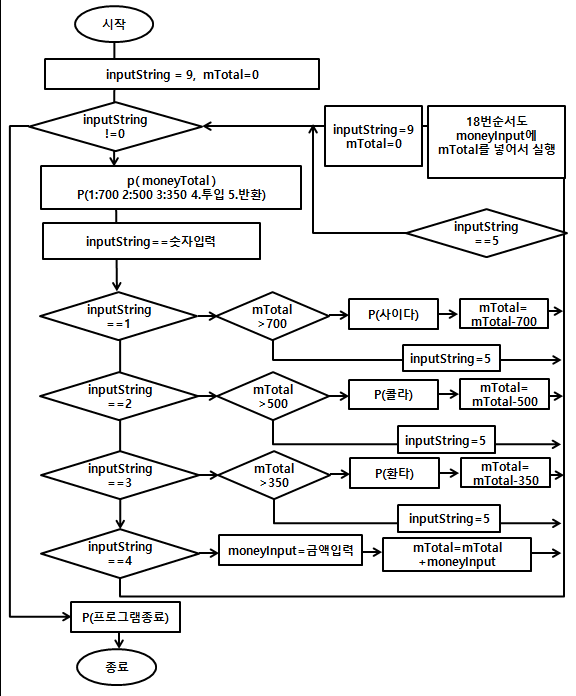
16. 피보나치 수열 10개를 순서대로 출력하는 순서도와 프로그램을 작성해 보자.

17. 은행 프로그램을 순서도와 프로그래밍 언어로 만들어보자.



18. 금액을 입력받아 천원 오백원 백원 오십원 짜리 잔돈으로 거슬러주는 프로그램과 순서도을 구현해 보자.

19. 실존하는 자판기와 동일하게 만들어 보자.



|  |
| --- |
| > 05. 배열 |

여러개의 데이터를 한번에 생성하는 것을 배열 이라고 한다. 변수는 5개의 저장 공간을 만들려면 a1=0, a2=0, a3=0, a4=0, a5=0 이런 식으로 일일이 선언 하여야 한다. 이런 번잡함을 해결하기 위해서 한번에 저장 공간을 만드는 방법이 제공되는데 이를 배열이라고 한다.

배열의 사용 방법은 배열명[인덱스]이렇게 기술한다. 배열명은 변수명이 변수들을 구분할때 사용 하듯이 배열을 구분하는데 사용한다.변수로 저장 공간 5개를 만드는 것이 a1=0, a2=0, a3=0, a4=0, a5=0와 같다면 배열로 저장공간 5개 만드는 방법은 a[5],a[]={0,0,0,0,0}와 같다. a는 배열명이고 5는 저장공간크기를 의미한다. 여기서 a는 변수명 처럼 본인이 원하는 이름을 넣으면 되고 5는 원하는 데이터양 만큼 선정하면 된다. 배열명[인덱스]에서 인덱스는 저장공간의 위치를 의미 한다. 인덱스는 0부터 시작한다는 것을 잊지 말자. 배열은 a[0]=0, a[1]=0, a[2]=0, a[3]=0, a[4]=0 같은 형태로 값을 넣을 수 있고 a[0], a[1], a[2], a[3], a[4] 형태로 값을 읽어 올 수 있다.

a1, a2, a3 등과 같은 경우는 저장 공간과 변수명이 1개 여서 a1에 1를 넣고 싶으면 a1=1 a2에 2를 넣고 싶으면 a2=2 이런 식으로 1:1 연결해서 접근 가능 하지만 배열 같은 경우에는 배열명이 1개 일 때도 여러 개의 저장 공간을 가진다. 같은 이름으로 여러 저장 공간에 접근하기 위해서 인덱스를 사용 하는데 a[인덱스] 즉 a[0], a[1], a[2]이런 식으로 접근 할 수 있다. 중요한 것은 인덱스는 0 부터 시작한다는 것이다. 첫번째 값에 접근하려면 a[0] 두번째 값에 접근하려면 a[1], 세번째 값에 접근하려면 a[2]하면 된다. a[12]와 같이 값을 가지고 있지 않은(할당 되지 않은) 인덱스로 값에 접근 하려 하면 에러가 발생 한다.(잘못된 결과가 발생한다.)

5개 저장공간을 가진 변수와 배열에서 첫번째 값을 가져오는 방법은 변수의 경우 a1, 배열의 경우 a[0]이라고 하면 된다. 세번째 값을 가져오는 방법은 변수의 경우 a3, 배열의 경우 a[2]이라고 하면 된다. 네번째 데이터의 값을 4로 변경하고 싶을 경우 변수의 경우 a4=4, 배열의 경우 a[3]=4이라 하면 된다.

a라는 배열에 5,2,3를 넣은 다음 : a[0]=5, a[1]=2, a[2]=3

변수b가 4일때 a배열의 인덱스[1]번을 b만큼 증가시킨 다음 : a[1]=a[1]+b

인덱스 0과 인덱스 1를 더해서 인덱스 2에 넣은 다음 : a[2]=a[0]+a[1]

인덱스 0,1,2를 더한 값을 a[3]에 넣은 다음 : a[3]=a[0]+a[1]+a[2]

a 배열의 모든 값의 합을 sum에 넣은 다음:sum=a[0]+a[1]+a[2]+a[3]

sum를 5로 나눈다음:sum=sum/5

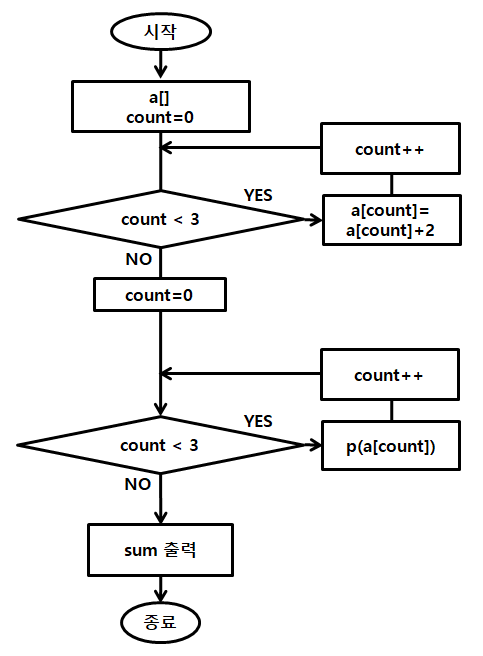
sum를 출력해 보자.

선언과 함께 배열의 값을 설정 하는 방법도 있다. a[]={1,2,3,4,5} 이렇게 하면 a[0]에 1, a[1]에 2, a[2]에 3, a[3]에 4, a[4]에 5가 들어 있는 것과 같은 결과 가 나올 것이다.

다음 문제들을 순서도와 프로그램으로 구현해 보자.

1. 인덱스 상수 숫자 대신에 변수 숫자를 넣을 수 있다. a[]={1,2,3,4,5}, b=1, p(a[0]), p(a[b]) 와 같이 사용할 수 있고 실행 결과는 1과 2가 출력 된다.

2. 배열 a에 1,2,3 을 넣은 후 배열의 값에 2를 더한 값으로 변경한 다음에 배열의 내용을 화면에 출력하는 순서도와 프로그램을 구현하시오.



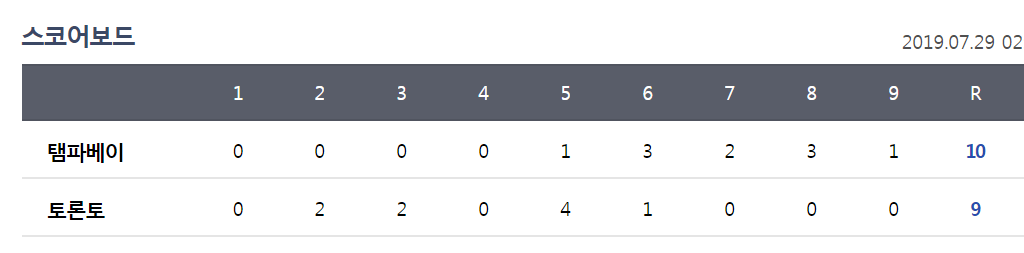
3. 다음을 순서도로 만들어 보자. 배열 a에 인덱스 0~5에 1부터 6까지 순서대로 숫자를 집어 넣은 다음에 배열의 내용을 더한 값을 sum에 저장하여 출력하는 순서도를 만들어 보자. 배열 초기화 방법을 사용하지 않고 만들어 보자.

4. 배열에 3의 배수를 넣는 순서도와 프로그램을 작성해 보자.

5. a[]={12,1,5,3,6,8,5,3}의 a배열에서 배열 안의 숫자가 짝수인 12,6,8 의 값을 더하는 순서도와 프로그램을 작성해 보자.

6. a[]={12,1,51,3,6,8,5}의 a배열에서 가장 큰 값과 가장 작은 값을 더하는 순서도와 프로그램을 작성해 보자.

7. 아래는 야구게임 전광판이다. 여러 변수들을 이용하여 게임이 진행되는 순으로 값들을 화면에 출력해보자.0,0,0,2,0,2,0,0,.....1,0,10,9 다음과 같이 출력되면 된다. 순서도와 프로그램을 작성해 보자.



상위 그림은 야구점수 전광판 이다. 상위 정보를 표현하는데 필요한 자료들이 어떠한 것들이 있는지 기술해 보자. 일단 팀파베이, 토론토는 야구 팀이다 두팀의 1~9회까지 점수를 저장할 배열 2개와 총합을 저장할 변수 2개가 있으면 야구점수 관련된 모든 데이터를 저장할 수 있을 거 같다. 물론 배열 하나에 모든 정보를 기록 할 수 있다. 0~9까지는 탬파베이 점수 10~19까지는 토론토 점수 해도 되지만 이렇게 된다면 배열을 사용하는 것이 불편해지므로 사용하기 편한 형태로는 두팀의 회수 정보배열2개와 총합변수 2개가 적당하다. 선언해 보자. team1[], team2[], team1TotalScoure=0, team2TotalScore=0 이해하기 쉬운 형태의 변수명을 사용해서 선언해 보았다. 배열 team1에 탬파베이 점수값을 넣어보자. 코드 줄수를 줄이기 위해서 배열을 선언하면 기본 값이 0 이라고 생각하고 진행하자. 5회에 1점을 얻었다 team1 배열에 이정보를 넣으려면 team1배열의 4번째 인덱스에 1를 넣으면된어 다음과 같이 하면된다. team1[4]=1 6회에 3점을 얻었다 team1[5]=3 team1[6]=2… 이런 식으로 진행해 나가면 된다. 토론토의 점수를 배열에 넣으려면 다른 배열 team2에 넣으면 된다. team2[1]=2, team2[2]=2 이런 식으로 넣으면 된다. 두팀의 총 취득한 점수를 어떻게 구할 것인지 생각해보자. 첫번째 팀의 경우 다음과 같이 team1의 인덱스 0 부터 8까지 모두 저장하면 된다. 이 방법이 복잡하다면 초기화 방법으로 각팀의 점수를 초기화 하자. team1TotalScoure=team1[0]+team1[1]+team1[2]+team1[3]+team1[4]+team1[5]+team1[6]+team1[7]+team1[8] 이렇게 하면 팀1의 득점한 총 점수를 출력할 수 있을 것이다. 지금 처럼 일일이 써서 넣을 수 도 있지만 다음과 같이 여러 줄로 처리할 수 있다.

team1TotalScoure=team1TotalScoure+team1[0]

team1TotalScoure=team1TotalScoure+team1[1]

team1TotalScoure=team1TotalScoure+team1[2]

team1TotalScoure=team1TotalScoure+team1[3]

team1TotalScoure=team1TotalScoure+team1[4]

team1TotalScoure=team1TotalScoure+team1[5]

team1TotalScoure=team1TotalScoure+team1[6]

team1TotalScoure=team1TotalScoure+team1[7]

team1TotalScoure=team1TotalScoure+team1[8]

team1TotalScoure=team1TotalScoure+team1[9]

여기에서 인덱스를 변수로 사용할 수 있다. 변수를 사용해서 다음과 같이 변경해 보자.

i=0

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

team1TotalScoure=team1TotalScoure+team1[i]

i++

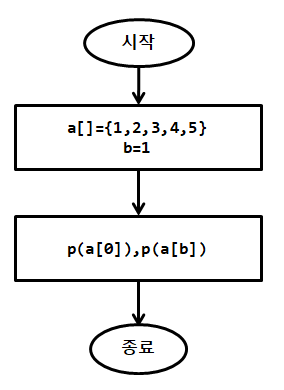
team1TotalScoure=team1TotalScoure+team1[i]

i++

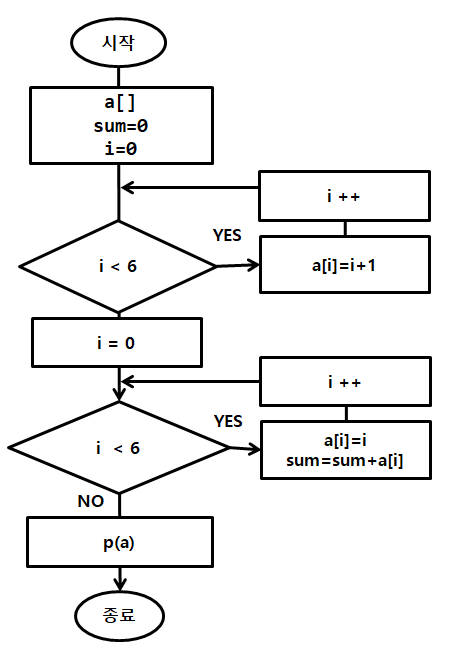
team1TotalScoure=team1TotalScoure+team1[i]

i++

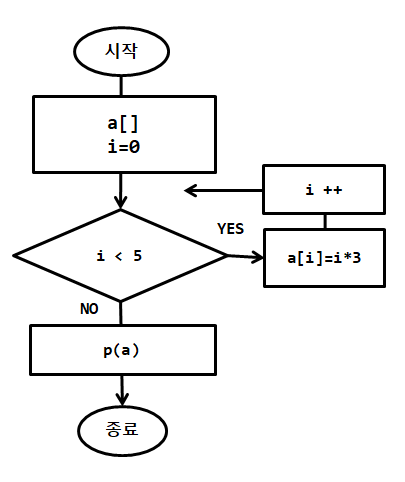
이렇게 변수를 사용하면 처음과 동일한 결과가 나올것이다. 이렇게 변경해서 사용하는 이유는 보면 알겠지만 다음 같은 부분이 반복되고 있어 team1TotalScoure = team1TotalScoure + team1[i] i++ 이 부분을 반복 시키면 더욱 간단한 형태로 알고리즘 순서도을 구현할 수 있다.

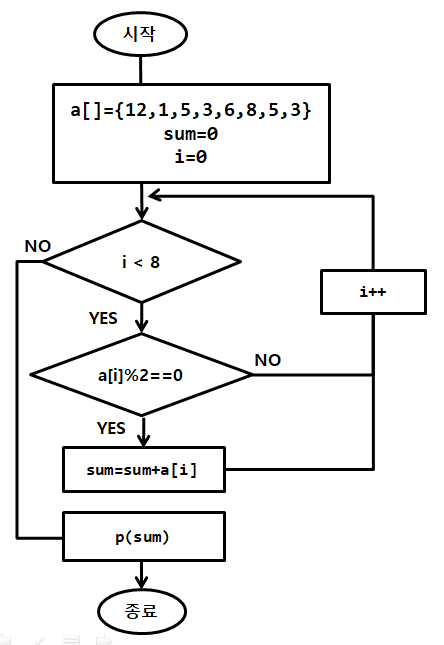
배열 순서도 답안

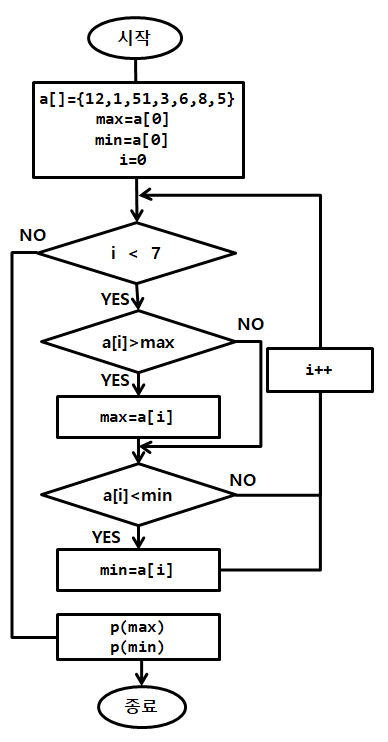
1. 인덱스 상수 숫자 대신에 변수 숫자를 넣을 수 있다. a[]={1,2,3,4,5}, b=1, p(a[0]), p(a[b]) 와 같이 사용할 수 있고 실행 결과는 1과 2가 출력 된다.

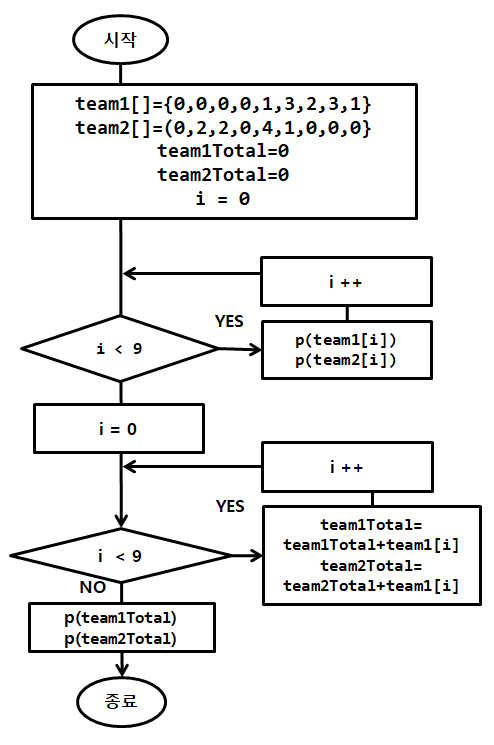


3. 다음을 순서도로 만들어 보자. 배열 a에 인덱스 0~5에 1부터 6까지 순서대로 숫자를 집어 넣은 다음에 배열의 내용을 더한 값을 sum에 저장하여 출력하는 순서도를 만들어 보자. 배열 초기화 방법을 사용하지 않고 만들어 보자.

4. 배열에 3의 배수를 넣는 순서도와 프로그램을 작성해 보자.

5. a[]={12,1,5,3,6,8,5,3}의 a배열에서 배열 안의 숫자가 짝수인 12,6,8 의 값을 더하는 순서도와 프로그램을 작성해 보자.

6. a[]={12,1,51,3,6,8,5}의 a배열에서 가장 큰 값과 가장 작은 값을 더하는 순서도와 프로그램을 작성해 보자.

7. 아래는 야구게임 전광판이다. 여러 변수들을 이용하여 게임이 진행되는 순으로 값들을 화면에 출력해보자. 0,0,0,2,0,2,0,0, .....1,0,10,9 다음과 같이 출력되면 된다. 순서도와 프로그램을 작성해 보자.

|  |
| --- |
| > 05. 블랙잭 순서도 |

어려울 수 있으니 이 챕터는 하지 않아도 된다. 재미로 보고 말자.

2명의 player가 블랙잭을 하는 프로그램을 구현하여 보자.

게임 방법은 트럼프 카드 52장을 2명의 player가 번갈아 가면서 1장씩 원하는 만큼의 카드를 받아 각자의 카드의 합이 21에 가까운 사람이 이기는 게임이다.

카드는 각자가 원하는 만큼 받을 수 있다. 각자 번갈아 가면서 카드를 받고 싶은지 확인해서 받기를 원하면 1장씩 받고 한 명이 더 이상 카드를 받길 원치 않으면 혼자서 카드를 받는다. 그리고 둘다 더 이상 카드를 받길 원하지 않으면 각자 받은 카드의 합이 21에 가까운 사람이 이긴다. 만약 카드를 받는 중 21일이 넘어가면 무조건 게임에서 패하게 되고 게임은 종료된다. 둘다 동시에 21이 넘어가게 되면 카드를 먼저 받은 사람이 지게된다.

카드별 카운트 해야할 숫자는 다르다 카드 A 의 경우에는 1과 11중 원하는 수를 카운트 한다. 2~10 카드는 카드 숫자 만큼 추가 한다. J,Q,K 카드는 10으로 계산 한다.

필요한 데이터를 확인해보자.

deck[] 카드 52장

CardShape[] {“스페이드”,”하트”,”다이아”,”나무”}

출력할 때 사용할 카드 모양 배열을 만들 었다.

CardNumber[] {“A”,”2”,”3”,”4”,”5”,”6”,”7”,”8”,”9”,”10”,”J”,”Q”,”K”}

출력할 때 사용할 카드 넘버를 배열로 만들 었다.

deckIndex 사용할 카드의 인덱스를 의미한다. 0이면 카드를 하나도

사용하지 않은 것이고 52이면 모두 사용한 것이다.

p1Deck[] 1번 플레이어가 취득한 카드

p2Deck[] 2번 플레이어가 취득한 카드

p1Flag 1번 플레이어가 카드 취득 여부

p2Flag 2번 플레이어가 카드 취득 여부

p1Index 1번 플레이어가 취득한 카드갯수

p2Index 2번 플레이어가 취득한 카드갯수

playState 게임 상태를 의미한다. “”, “p1승리”, “p2승리”, “무승부”

게임중 화면 1

p1님 카드 받겠습니까? 0=no,1=yes 1 <--사용자 입력

p2님 카드 받겠습니까? 0=no,1=yes 1 <--사용자 입력

>>카드 현황

P1:하트9, p1총점>>9

P2:스페이드A, p2총점>>11

게임중 화면 2

p1님 카드 받겠습니까? 0=no,1=yes 1<--사용자입력

p2님 카드 받겠습니까? 0=no,1=yes 1<--사용자입력

>>카드 현황

P1:하트9,클로버5, p1총점>>14

P2:스페이드A,다이아7, p2총점>>18

게임중 화면 3

p1님 카드 받겠습니까? 0=no,1=yes 0<--사용자입력

p2님 카드 받겠습니까? 0=no,1=yes 1<--사용자입력

>>카드 현황

P1:하트9,클로버5, p1총점>>14

P2:스페이드A,다이아7,스페이드8, p2총점>>16

게임중 화면 4

p2님 카드 받겠습니까? 0=no,1=yes

0

두 플레이어가 카드 수령을 중지하였습니다.

게임종료후 화면

------------------

>>카드 현황

P1:하트9,클로버5,

p1총점>>14

P2:스페이드A,다이아7,스페이드8,

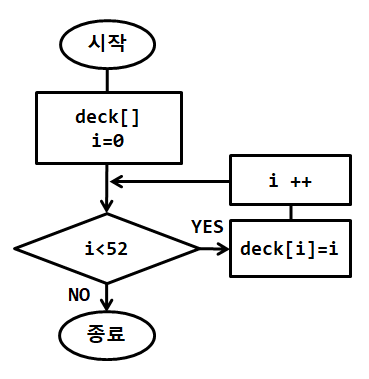
p2총점>>16

p2승리

동작순서

1. 카드 52장을 만든다.
   1. 0~51까지 고유한 번호를 넣어 카드를 구분한다.
2. 카드를 섞는다.
   1. deck배열의 내용을 섞는다.
3. 원하는 플레이어가 1장씩 번갈아 받는다.
   1. 카드를 준다는 것은 deck 배열에서 deckIndex의 위치에 카드 값을 배열 p1Deck, p2Deck에 넣어주면 된다.
   2. 카드를 주고 나면 deckIndex을 하나 증가시켜 다음 사용할 카드를 가리키고 p1Index나 p2Index을 증가하여 다음 넣을 곳을 가리킨다.
4. 카드를 받을지 여부에 플래그 p1Flag, p2Flag를 사용한다.
   1. 더이상 카드를 받지 않는 플레이어는 카드 받을 지 여부를 물어 보지 않고 카드를 주지 않는다.
5. 카드를 받을 때 마다 21이 넘으면 게임을 종료하고 게임에 진것이 된다.
6. 둘다 카드를 더이상 받지 않으면 게임을 종료하고 승패를 종료한다.
7. 카드 게임중 플레이어가 가지고 있는 카드를 주고나서 혹은 게임이 끝나고 나서 화면을 보여 준다.
   1. 게임화면 만들때 두명의 플레이어가 가진 모든 카드를 보여준다.
8. 카드 게임 종료전 게임에 승패를 화면에 출력 한다.

카드생성

순서도에서 사용하는 용어를 간단히 설명해보고 순서도를 확인해보자.

deck[]: deck배열: 전체 카드 배열: 게임을 하려면 전체 카드를 저장해 놓은 배열이 필요하다.

deck[i]: 카드의 식별자: deck배열에 들어 있는 카드 중 deck배열의 i번째 들어 있는 하나의 카드 식별자로 0~51사이의 숫자가 된다: 52장의 카드를 구분하는 숫자.

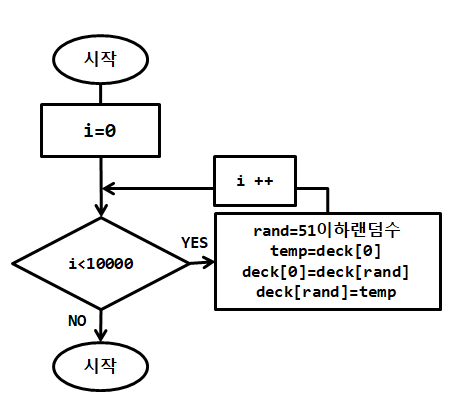
i: 카드의 위치 혹은 카드의 식별자: 카드식별자 값 혹은 deck배열에서 몇번째 위치하는 지에 대한 정보: 카드 식별자 혹은 카드를 쌓아 놓으면 층이 생기는데 결국 몇번째 층에 카드를 읽어올 것인지 결정하는 카드의 위치, 즉 deck배열의 인덱스로 사용한다.

deck[i]=i에서 앞에서 사용한 deck[i]의 i는 배열의 인덱스 카드의 위치를 나타낼때 사용한 것이고 뒤에 있는 i는 카드의 식별자 즉, 스페이드 1인지 2인지 를 구분할 수 있는 식별자이다. deck[i]에서 i는 카드의 위치를 나타내고, deck[i]는 i위치에 들어 있는 카드의 식별자를 의미한다.

프로그램에서의 카드를 생성한다는 것은 모든 카드를 구분할 수 있도록 만들면 된다. 0부터 51번까지의 번호를 넣어두고 0은 스페이드A, 1은 스페이드2, 2는 스페이드3, ... 13은 크로버A, 14는 크로버2…… 이런 식으로 0~51를 가지고 카드를 식별 할 예정 이다. 이러한 개념으로 카드1벌을 프로그램으로 만든다면 0~51까지의 숫자를 넣은 배열을 만들면 카드 1벌이 만들어진 것이고 1벌을 deck이라 한다. i를 0~51까지 반복해서 deck[0] ~ deck[51]까지 배열에 0~51까지의 식별자를 순서대로 넣으면 식별 가능한 deck[]배열 카드 52장을 넣을 수 있게 된다. deck[i]=i에서의 카드 위치인 배열인덱스 i와 카드 식별값i를 명확하게 구분해야 한다.

배열인덱스 i는 카드덱에서의 위치를 의미한다. 카드가 쌓여 있다고 생각해보자. 52장이 층을 이루고 있을 것이고 맨 상단의 카드가 배열의 인덱스 i 값 0에 해당 할 것이다. 그럼 실제로 맨위에 있는 카드가 무슨 카드인가? 그 카드에 들어 있는 카드값 (카드의 식별자)가 무엇인지 알아야 할것이다. deck[i]가 카드의 식별자이다.

상의 순서도를 차례대로 따라가 보면 카드 1벌이 만들어질 것이다.

카드섞기

deck[]: deck배열: 전체 카드 배열: 게임을 하려면 전체 카드를 저장해 놓은 배열이 필요하다.

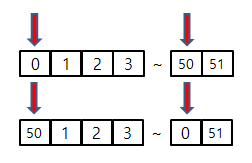
temp: 임시로 카드식별자를 저장할 공간: deck배열의 특정 위치 2개를 교환하려면 임시로 저장할 공간이 필요하다.

rand:51이하의 랜덤수: 카드의 0번 인덱스와 교환할 카드 인덱스를 구할때 사용한다.

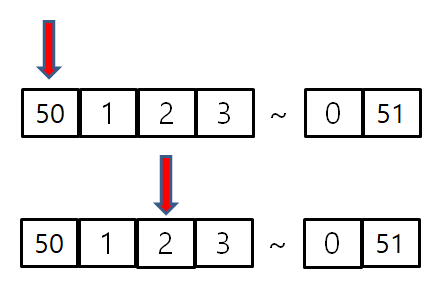
상위 순서도는 deck배열의 내용을 섞는 작업을 하는 것이다. 섞지 않으면 배열에 숫자가 들어 있는데 정렬되어 있다. 카드를 게임에 사용 하려면 카드를 섞어야 한다. 인덱스 0부터 순서대로 읽어 오면 다음 나올 카드를 누구나 알 수 있다.

프로그램에서 카드를 섞는다는 의미는 배열의 내용를 교환 한다는 의미이다.

잘 생각해보면 카드를 섞으면 해당 층의 카드 정보가 바뀔 뿐이지 카드 층 정보가 바뀌지 않는다. 카드를 쌓아 놓았 을 때 맨 위쪽을 0번 층 맨아래 층을 51번 층이라고 이야기 하였다면 카드를 섞고나서 카드를 쌓아 놓는다면 여전이 맨 위쪽층은 0층, 아래층은 51층이 된다. 결국 층정보는 변하지 않고 해당 층의 카드 정보만 변경된다.

카드 내용을 섞는 방법은 여러가지 있는데 왼쪽 이미지 처럼 인덱스 0번과 50번의 데이터를 교환 하는 것이다. 이렇게 교환이 되면 배열의 내용은 손실없이 섞이게 된다.

이때 이 50번 대신에 랜덤한 수를 구해서 0과 여러번 교환 하면 내용이 섞이게 된다. 상위 순서도는 카드를 만번 섞는 방법을 순서도로 만든 것이다.

deck에서 카드 한장 사용하기

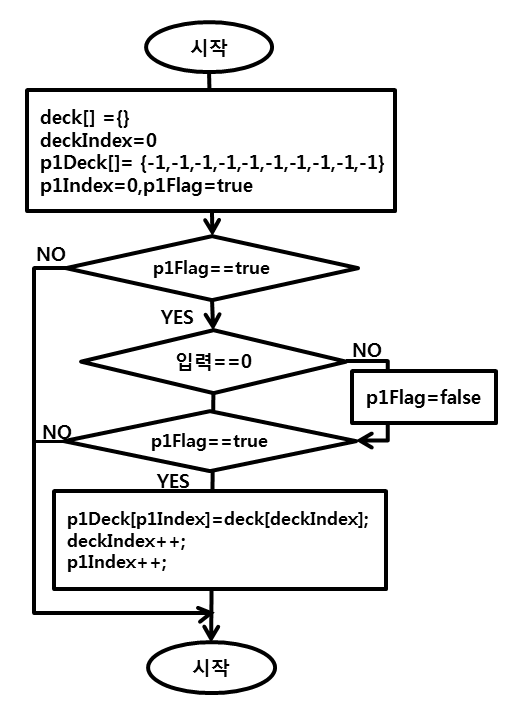
프로그램에서 카드덱에서 카드 한장을 받는다는 의미는 카드 덱 배열에서 사용한 카드 부분과 사용하지 않은 카드 부분, 다음 사용할 카드를 확인 할 수 있다는 이야기 이다. 프로그램에서 deckIndex를 가지고 확인 할 수 있다. deckIndex가 가지고 있는 수를 기준으로 deckIndex보다 작은 수의 deck배열의 인덱스는 이미 사용한 카드이고 deckIndex와 같은 위치의 deck배열에 인덱스 값은 다음에 사용할 카드가 되고, deck배열에서 deckIndex보다 큰 인덱스에 존재하는 값들은 다음에 사용할 카드들이다. 상위 이미지에서 위쪽은 게임하기전 최초 카드를 만들었을때 상황을 표현한 것인데 빨간 화살표 부분이 배열에서 다음 사용할 카드를 가르키는 deckIndex고 현상황은 deckIndex가 0 이여서 카드가 섞여진 deck배열 의 맨처음을 가리키고 있다. deckIndex가 0 이여서 deckIndex보다 작은 deck배열의 인덱스가 존재하지 않으므로 사용한 카드가 없고 deckIndex가 0이므로 deck배열의 인덱스가 0인 0번째 카드 50이 다음에 사용 할 카드 이고 1, 2, 3~, 0, 51이 아직 사용하지 않은 카드이다.

여기서 카드를 한장 사용 한다고 생각해보자. 현재 deckIndex가 0 이므로 deck 배열의 인덱스 0인 카드를 사용 할 것 이다. 카드를 사용하고 나면 다음 두가지 작업을 해야 한다. 첫번째는 deck배열의 0번 인덱스의 카드를 deckIndex보다 작게 만들어 사용한 카드로 만들어야 한다. 두번째 deckIndex의 값을 아직 사용하지 않은 deck배열의 인덱스로 바꿔주는 작업을 해야한다. 이 두가지 조건을 deckIndex++로 만족 시킬 수 있다.

카드 하나를 사용하여 deckIndex가 하나 증가하면 deckIndex는 1이 되고 이보다 작은 deck배열의 인덱스는 0이 존재하고 deckIndex보다 작은 인덱스이니 실제로 사용한 카드 이다. deckIndex가 1이되면 deck배열에서 인덱스1인 카드가 다음에 사용할 카드가 되고 deckIndex보다 큰 deck배열의 인덱스 값들은 아직 사용하지 않은 카드가 된다.

결국 카드 deck배열에서 카드 한장씩 읽어오는 방법은 deckIndex와 동일한 deck배열의 카드 식별자을 읽어온다음 deckIndex를 하나 증가시켜 주면된다.

위쪽 이미지에서 아래부분은 deck배열의 카드를 2장 사용하여 deckIndex가 2가되고 사용한 카드가 50,1 다음 사용할 카드가 2, 아직 사용하지 않은 카드가 3 ~ 0,51 인 상태를 보여주고 있는 것이다.

사용자가 카드 한장 받기

p1Deck[]: p1 플레이어의 카드 배열: 게임중 플레이어가 받은 카드를 의미한다.

p1Index: p1 플레이어가 가지고 있는 카드 수를 의미 한다.

p1Falg: p1 플레이어가 카드를 받을지 여부를 결정하는 플래그이다.

deck배열에서 카드 한장을 사용하면 사용자가 카드 한장을 받는 작업을 프로그 램으로 구현하여야 한다. 사용자가 카드 한장을 받는 다는 것은 p1Deck 배열에 카드를 넣는것을 의미한다.

여기 순서도에서는 카드 한장을 받는 것으로 구현 되어 있지만 실질적으로 사용자가 지속적으로 카드를 여러장 받아야 한다. 한장 받는 작업을 여러번 하면된다.

일단 사용자가 이전에 카드를 그만 받기로 결정 하지 않았다면(p1Flag값이 true이면 카드를 계속 받기로 한 것이다.), 카드를 그만 받을 것인지 계속 받을 것인지 물어 보고 사용자로 부터 입력을 받는다. 사용자가 1을 입력하면 계속 받을 예정이어서 p1Flag값을 변경하지 않고 true로 남겨두면 되고 0을 입력하면 더이상 카드를 받지 않을 예정이어서 p1Flag값을 false로 변경하여 더이상 카드 받는 작업을 그만 두게 한다.

사용자가 카드를 계속 받기를 원하면 deck배열에서 카드 한 장을 꺼내 p1플레이어의 카드를 보관하는 배열 p1Deck[]에 넣어 주면 된다. p1Deck[]에 -1를 10개 넣은 이유는 p1플레이어가 카드 10개를 받을 수 있도록 미리 세팅한 것이다.

deck에서 카드 한장을 p1Deck[]에 넣는 방법은 deck 배열에서 다음에 사용할 카드 인덱스가 deckIndex이고 p1Deck[] 배열에서 p1플레이어가 받은 카드를 저장할 인덱스 번호가 p1Index이므로 p1Deck[p1Index]=deck[deckIndex]과 같은 방법으로 deck배열 에서 p1Deck배열로 카드 식별자를 복사하면 된다. 카드를 복사한 다음에는 deck에서 카드 한장을 사용 하였 으므로 deckIndex++를 이용하여 다음 사용할 카드로 이동 하여야 한다. p1Deck[]배열에 카드 한장이 들어 갔으므로 p1Index++를 이용하여 다른 카드를 저장 할 수 있도록 다음 저장 공간을 가리켜야 한다.

카드출력

0~51까지의 카드 식별자로 사용자가 알아보기 쉬운 스페이드A, 1은 스페이드2, 2는 스페이드3, ... 13은 클로버A, 14는 클로버2… 형태로 출력 할 수 있어야 한다.

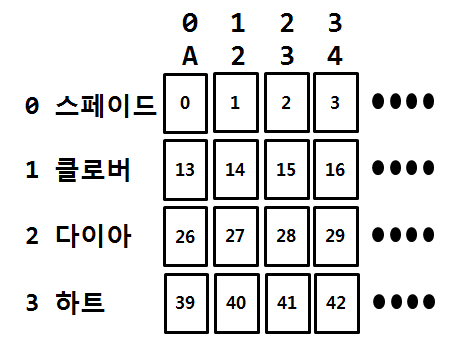
deck배열이나 p1Deck배열에는 0부터 51번까지의 카드 식별 번호가 들어 있을뿐 0은 스페이드A, 1은 스페이드2, 2는 스페이드3, ... 13은 클로버A, 14는 클로버2…… 이런 카드 정보는 들어 있지 않다. 0~51까지의 숫자를 가지고 스페이드A ~ 하트 k 까지 사용자가 알아보기 쉬운 형태로 화면에 출력 할 수 있어야 한다.

cardShape[]: 스페이드와 같은 카드 모양 종류를 문자열로 저장하는 배열이다.

cardNumber[]: A,1,2..와 같은카드 숫자 모양을 문자열로 저장하는 배열이다.

p1Deck[]:p1플레이어가 가지고 있는 카드묶음이다.

p1Index: p1플레이어가 가지고 있는 카드 개수이다.

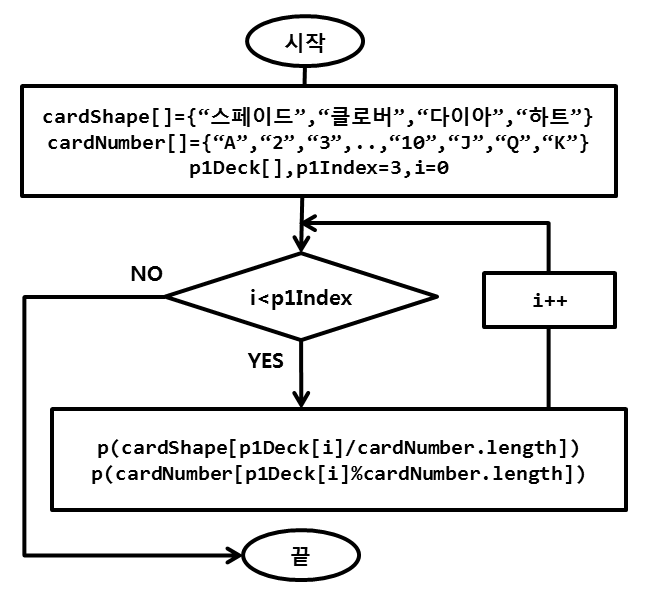
0~51번호를 가지고있는 52장의 카드를 잘 생각해 보면 4가지 모양을 13장씩 4등분 하여 가지고 있다. 0~12는 스페이드, 13~25는 크로버, 26~38는 다이아, 39~51는 하트가 된다. cardShape 배열에는 스페이드, 클로버, 다이아, 하트가 문자열로 들어가 있어 cardShape[0]를 출력하면 스페이드가 출력 된다. 카드 식별 번호가 0~12인 카드는 스페이드로 출력 해야 하고 cardShape[0]이 스페이드 문자열 이므로 0~12숫자를 0으로 만들면 문자열 스페이드로 cardShape를 이용해서 출력 할 수 있다. 0~12 사이의 숫자를 0으로 만들려면 13으로 나눈 몫을 구하면된다. 정수의 계산에서 정수/정수는 정수가 되어 소수점 부분이 없어진다. 13~25사이 숫자도 1로 만들려면 cardShape 배열을 이용하여 클로버로 출력 할 수 있고 13~25를 13로 나누면 1이된다. 26~38를 13으로 나누면 2가 되고 39~51를 13으로 나누면 3이 된다. 결론적으로 카드 식별자를 13으로 나눈 몫을 cardShpe배열의 인덱스로 사용하면 카드의 모양을 문자열로 출력할 수있다. 사용자가 가지고 있는 첫번째 카드의 카드 식별 번호는 p1Deck[0]이고 이식별 번호를 13으로 나누면 실제 카드의 모양이된다. 13은 한 모양의 카드가 표현할 수 있는 숫자 모양 개수이므로 cardNumber배열의 개수와 동일하여 13 대신에 cardNumber.length를 사용하였다. p1Deck[0]/cardNumber.length 이렇게 하면 

p1플레이어가 가지고있는 1번째 카드식별번호/13이므로 0~3까지의 숫자중 하나가 만들어져 cardShape배열의 인덱스로 사용되어 스페이스인지,클로버인지.. 화면에 출력 할 수 있다. 결국 다음을 실행하면 p(cardShape[p1Deck[0]/cardNumber.length]) 결과로 스페이드,클로버,... 처럼 해당 카드에 카드 모양이 출력된다.

다음은 카드 식별 번호로 해당 카드 넘버를 출력하는 방법을 생각해보자. 카드넘버는 A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K 13개가 있다. 상위 이미지를 살펴보면 카드 식별 번호를 13으로 나눈 나머지가 카드 번호가 된다. 0,13,26,39를 13으로 나누면 0이되고 cardNumber배열에 인덱스를 0으로 넣으면 cardNumber[0]이되고 이것은 A에 해당한다. 나머지가 1이면 2가 되고 3이면 4 가되는 것을 확인 할 수 있다. 결론은 p1플레이어가 가지고 있는 첫번째 카드의 숫자를 문자열로 출력하고 싶다면 p(cardNumber[p1Deck[0]%cardNumber.length])이라고 하면 된다. p1플레이어의 첫번째 카드모양과 카드 번호를 출력하고 싶다면 아래처럼 2줄 써주면 될것이다.

p(cardShape[p1Deck[0]/cardNumber.length]) p(cardNumber[p1Deck[0]%cardNumber.length])

아래 순서도는 플레이어가 가지고 있는 카드 3장을 모두 출력하는 순서도이다. p1Deck 배열에 들어 있는 카드 식별 번호를 이용해서 반복하여 카드별로 모양과 번호를 출력하는 순서도 이다.



점수계산

플레이어가 가지고 있는 카드를 모두 더하여 점수를 계산하는 순서도이다.

p1Index: p1플레이어가 가지고 있는 카드 개수이다.

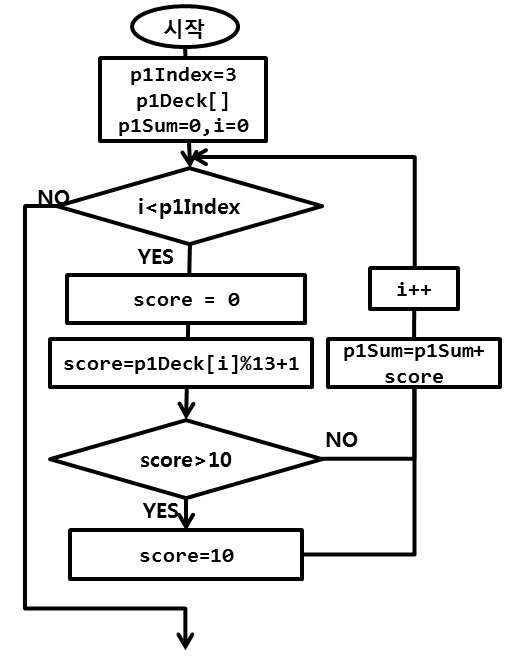
p1Deck: p1플레이어가 가지고 있는 카드 식별자 배열이다.

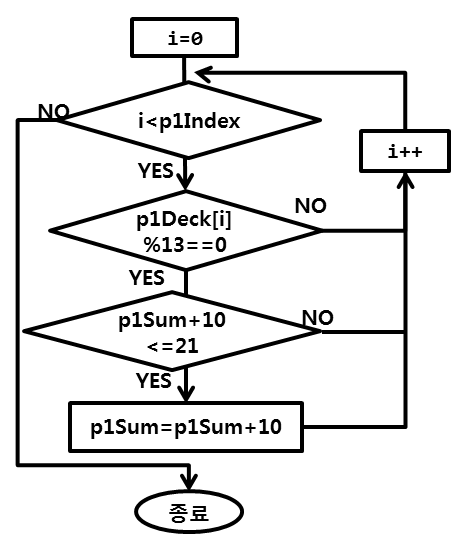
p1Sum: p1플레이어가 가지고 있는 카드들의 총합을 구하기 위해서 누적하는 변수이다.

scoure: 카드 1개의 점수이다.

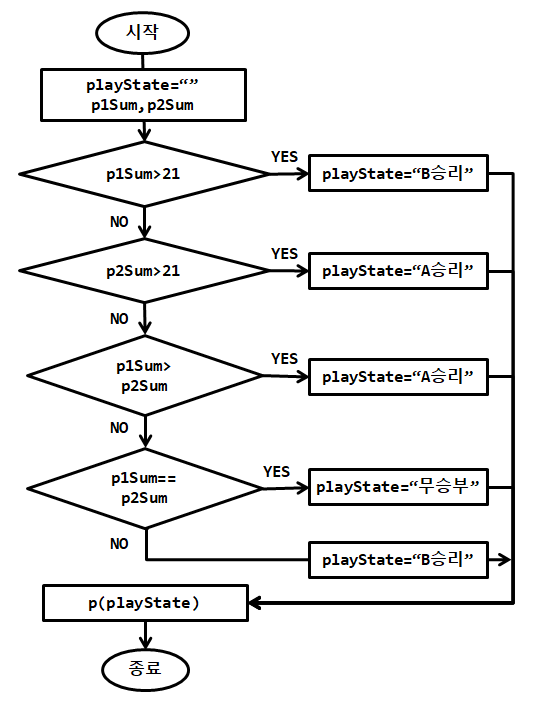
카드별 카운트 해야할 숫자는 다르다 카드 A 의 경우에는 1과 11중 원하는 수를 카운트 한다. 2~10 카드는 카드 숫자 만큼 추가 한다. J,Q,K 카드는 10으로 계산 한다.

일단 A같은 경우는 1과 11로 원하는 수를 선택해서 구할 수 있지만 1로 생각하고 계산한 다음 차후 설명할 예정이다. 카드식별자숫자%13한 결과는 0,1,2,3,4… 일때 A,2,3,4,5… 카드에 해당하고 카드점수는 카드식별자숫자%13한 결과에 +1한 것과 같아야 한다. 아래 순서도를 보면 i<p1Index로 p1플레이어가 가지고 있는 카드수 만큼 반복하고 있고 카드 한장을 score로 선언한 다음 0으로 초기화한후 사용자가 가지고 있는 p1Deck 배열에서 카드식별자를 읽어와 사용자가 가지고 있는 카드 1장의

카드 식별자 숫자를 읽어와 (p1Deck[i]) %13으로 나눈후 +1를 해주면 p1플레이어가 가지고 있는 한장의 카드 점수를 알수 있다. J,Q,K카드는 카드 식별 숫자를 %13으로 나누면 11, 12, 13이되고 이들 카드를 10 으로 계산하기로 되어 있으므로 만약 카드 점수인 scour가 10을 넘어간다면 11, 12, 13 인 경우 이므로 scour를 10으로 변경하여 천체 카드점수를 저장하는 p1Sum변수에 넣어주고 다음 카드를 합산하기 위하여 i를 하나 증가 시켜주고 i<p1Index로 p1플레이어 가 가지고 있는 카드 수만큼 반복 하여 p1플레이어가 가지고 있는 전체 카드 점수를 더한 p1Sum 값을 구할 수 있다. 총합산이 구해지면 전체 카드 점수 p1Sum값을 구한 것처럼 

보이지만 A 가 상황에 따라 1 또는 11로 변경 될 수 있다는 부분을 아직 적용하지 않았다. 왼쪽 순서도는 상황에 따라 A를 11로 변경해서 p1플레이어의 총합산 p1Sum에 적용하는 순서도이다.

카드를 처음부터 순회 하기 위해서 i를 0으로 다시 초기화 하였고 i<p1Index 를 이용하여 카드 수만큼 반복하였다. p1Deck[i]%13==0 부분은 카드 식별자 숫자가 0 인지 즉 A 인지 확인 하는 방법이다. A가 아니라면 아무런 작업을 할 필요가 없고 A라면 이미 1이 더해져 있는 상태여서 A를 11로 처리하고 싶다면 10만 더해주면 되는데 게임상 21이 넘으면 게임에 진것이 되므로 더한 후의 값이 21이 넘는 지 확인하여 21이하라면 전체합을 나타내는 p1Sum변수에 10을 더해 주면 A를 11로 변경한 것이 되고 21이 넘는다면 A를 1로 적용한 상태로 남겨 두면 된다. 이런 식으로 i를 하나 증가 시켜 p1Deck[i]에 또 다른 A가 있는지 p1Deck[i]배열을 탐색하여 A가 있을때마다 상위와 같은 작업을 반복하면 A를 1 또는 11로 적용한 카드의 총점수인 p1Sum이 나온다.

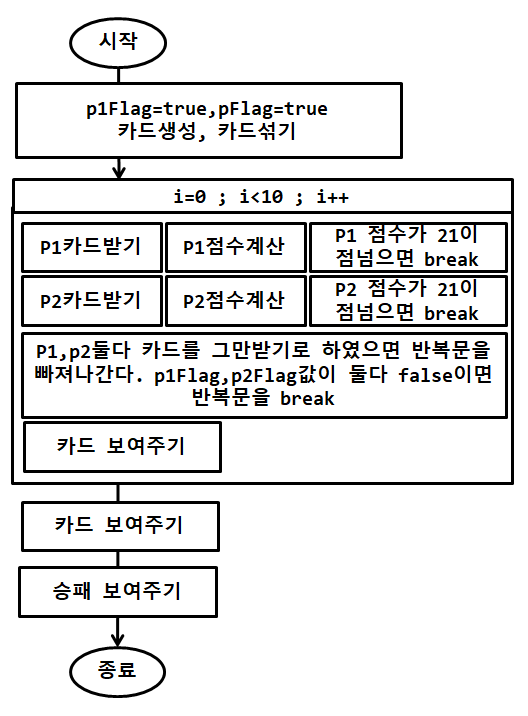
승패출력

p1Sum과 p2Sum은 p1,p2 플레이어가 가지고 있는 카드의 총합이다. playState는 두명의 플레이어중 누가 이겼는지 졌는지 문자열 로 만들어 저장 하는 변수이다. 게임 룰이 21를 넘으면 게임에서 지기 때문에 둘다 21이 넘으면 B가 승리하는 것으로 처리하였다.

게임진행

p1Flag는 p1플레이어가 카드를 계속 받을 것인지 받지 않을 것인지 결정하는 플래그 값이다. true이면 카드를 받는다.

p2Flag는 p1플레이어가 카드를 계속 받을 것인지 받지 않을 것인지 결정하는 플래그 값이다. true이면 카드를 받는다.

i=0;i<10;i++은 박스 부분을 10번 반복하라는 이야기이다. 

게임을 시작하면 카드생성과 카드섞이를 한 다음 p1플레이어가 카드 받을지를 경정하고 카드를 받으면 받은 카드의 점수를 계산한 다음 카드점수가 21이 넘으면 반복문을 빠져나가 게임을 종료하고 그렇지 않으면 p2플레이어가 카드 받는 작업을 진행한다. 이작업은 최대 10번 반복할 수 있고 중간에 둘다 카드 받는 작업을 더이상 원하지 않으면 보유한 카드를 보여주고 승패를 보여주고 게임을 종료한다.

