공통항목

1. 팩토리얼 연산

Flowgorithm

Raptor

1. Salary (Redirect\_output)
   1. 특징
      1. exporting data
      2. 중복 항목을 인식하지 못함. (ex. James: $100, James: $200 – 업데이트가 아닌 새로운 필드를 무조건적으로 형성함)
      3. 시스템이 접근 가능한 파일들만 가능한 듯 (.txt, .csv 가능, Docx 불가)
2. readFile (Redirect\_input)
   1. 특징
      1. 한 줄 씩 읽어 들임 (seperated by newline character)
3. readFileAndModifyContent
   1. 특징
      1. 2D Array가능 – strong type reference, reference 0 is null (must start at 1)
         1. 기존 프로그래밍 언어와 다른 부분
         2. If db[] is not an input paramter for a procedure, db[] is dynamically allocated in readFile procedure.
         3. **Limitation:** Length\_of can only take string or 1D array
         4. 논리는 완벽. &&, || 같은 경우 (T || F = true, F || T = false)   
            ex) if (db[i,1] = “1” || i > length) 같은 경우 첫 번째 조건이 false이기 때문에 두번째 조건을 체크하기 전에 false로 떨어짐.
4. Graph
   1. 특징
      1. Lot of defined methods should be used
      2. Open\_Graph\_Window(width, height)
      3. Set\_Window\_Title(string)
      4. Load\_Bitmap(path) -> this is an assignment (bitmap = Load\_Bitmap(path)
      5. draw\_bitmap(variable, cx, cy, width, height) -> cx, cy is top left corner although the canvas origin is left bottom corner with coordinat of (1, 1)
5. Pentool
   1. 특징
      1. Since the actual drawing part is implemented through a loop there is some delay in the pen.
      2. Raptor의 특징인지 아직 확실하지 않으나, 조건문(다이아몬드)에서 NO만 루프를 돌 수 있는 조건은 한정적인 것 같음. 특히 event-driven procedure에서 마우스 클릭시 procedure를 돌게 하고 싶은데 이것이 불가능.
6. Pentool(smooth\_working)
   1. regdedit -> HKEY\_LOCAL\_MACHINE -> SOFTWARE -> RAPTOR -> new string “reverse\_loop\_logic” = true.
   2. Or much simpler version just tell prefix the if statement in the loop with “not”
7. averageCalc
   1. simple average calculator program which gets number of subjects as user input
8. seatAssignment
   1. 특징
      1. 전체 학생 수와 원하는 row, col 수를 받아서 분단 형태로 array 출력
      2. 전체 학생 번호를 serialNum이라는 배열로 만들어서 (1 ~ 전체학생수)가 한 번씩 나올 수 있게 해줌
      3. Limitation: 현재는 총 학생수에 상관없이 array dimension만큼 무조건적으로 생성됨**. B-ii참조**.
   2. 느낀점
      1. Nested loop가 많아 질 수 록 (중첩 이상) 플로우차트를 알아보기 어려워짐
      2. **중간에 특정 조건이 만족되면 return할 수 있는 기능이 없어서 제한적임**
9. rockPapSci
   1. 특징
      1. 그냥 가위바위보 프로그램
      2. 기존에 있는 가위바위보 프로그램은 실제 사람이 하는 가위바위보 논리를 하드코딩해 놓은 것이 대부분임. 그래서 더 좋은 로직은 없을까 생각해 보다가 여기까지 왔는데 기존 하드코딩의 형태를 벗어나기는 어려웠음. 다만 어레이를 통한 데이터 구조를 만들어 놓고 이를 reference해서 1,2,3을 가위, 바위, 보에 대응시키는 논리 구조임.
      3. 그래도 쓸데없이 복잡하다고 생각됨. Refactoring 필요.
10. bankingAdvanced
    1. 특징
       1. OOP 코딩이 가능한 부분
    2. Class

|  |  |
| --- | --- |
| Fields | Methods |
| Constructor: Customer(int, int) {} 🡪 creates an account to hold info | |
| Int accountNum | Void changePassword() |
| Int savings | Void getSavings() |
| Int password | Void getAccountNum() |
| String name |  |

* + 1. Customer
    2. Bank

|  |  |
| --- | --- |
| Fields | Methods |
| Constructor: bank() {} 🡪 creates a bank for customers | |
| Customer[] customers | Int setAccountNum |
| Int totalNum | Void addCustomer(customer) |
|  | Void getTotalNum(customer[]) |
|  | Void deposit(customer, int) |
|  | Void withdraw(customer, int) |
|  | Void getAccount(string) |
|  | Void checkPassword(customer) |
|  | Void printAccountInfo(customer) |

* + 1. System

|  |  |
| --- | --- |
| Fields | Methods |
| Constructor: system() {} 🡪 creates a system that banks can run | |
|  | Void runSystem(bank) |
|  |  |

NotTestedFunc – not implemented in this flowchart

* 1. Specs
     1. This program is written for in perspective of an ATM or a bank teller. The user can create a client, manipulate a client’s info with correct information(client name and password).
     2. An individual user cannot access other user’s information
  2. ToDo
     1. Money transfer between two accounts

1. OOP (<https://www.youtube.com/watch?v=PQ9JMkwXgmE>)
   1. <https://www.youtube.com/watch?v=h71SCszLvXc>