**psh010209@gc.gachon.ac.kr, 010-7661-7035**

**Sanghee Park**

# **개발 과제 #1:**

|  |  |
| --- | --- |
|  | **Categorize input via age/hobby** |

**Requirements**

* **Step 1: Understand the problem - 간략한 핵심 요구 조건/작업 설명/예시 작성**

**Categorize personal.txt via age and hobby**

* **Step 2: Outline a solution – 간단한 logic설명**

**Check all file for category/age,**

**For all category/age, check if it is same, then write to file.**

* **Step 3: Form a program structure – 프로그램 전체 구조 요약**

**Ask user which do you want to group.  
then call the function for the specific function.  
  
And check all file for category/age,  
For all category/age, check it is same, and write.**

* **Step 4: write a pseudo coode – 수도코드 작성**

**Ask user to whether category or age it wants.**

**Read file and check for category/age**

**For read category/age,  
check all file and find matching category,  
if found, write to output file.**

## **SOURCE CODE with comments**

|  |
| --- |
| *#include* <stdio.h>  *#include* <stdlib.h>  *#include* <string.h>  */\* common \*/*  *// open file, when it failed, exit the program*  FILE \*safeFileOpen(char \*fileName, char\* fileMode) {  FILE \*fp = fopen(fileName, fileMode);  *if* (fp == NULL) {  printf("Failed to open %s! Check the file permission or Disk S.M.A.R.T.!\n", fileName);  exit(1);  }  *return* fp;  }  *// malloc but when it failed, exit the program*  void \*safeMemoryAllocation(size\_t size) {  void \*memAddr = (void \*)malloc(size);  *if* (memAddr == NULL) {  printf("Failed to allocate memory! Check is it out of memory!");  exit(1);  }  *return* memAddr;  }  *// Check how many line return characters are found in the file*  int howManyLines(FILE \*input) {  *// current location of file*  long a = ftell(input);  *// files have at least 1 lines*  int count = 1;  *// go to beginning*  fseek(input, 0, SEEK\_SET);  *// for all characters*  *for* (char c = 0; !feof(input); c=getc(input)) {  *// if line return increas 1*  *if* (c == '\n') {  count++;  }  }  *// to the previous location*  fseek(input, a, SEEK\_SET);  *return* count;  }  *// free all dynamically allocated double pointer*  void freeAll(void \*\*ptr, int length) {  *for* (int i = 0; i < length; i++) {  free(ptr[i]);  }  free(ptr);  }  */\* age \*/*  *// get ages from the file.*  int \*getAges(FILE \*input) {  *// save location temporaril;y*  long loc = ftell(input);    *// go to beginning.*  fseek(input, 0, SEEK\_SET);  *// get lines*  int lines = howManyLines(input);  *// get ages array*  int \*ages = (int \*)safeMemoryAllocation(sizeof(int) \* lines);    *// initalize*  *for* (int i = 0; i < lines; i++) {  ages[i] = -1;  }  *// temporary variables*  char name[10] = "";  int age = 0;  char hobby[10] = "";  *// current Age index to save at ages array*  int currentAgesIndex = 0;  *// for each lines*  *for* (fscanf(input, "%s %d %s\n", name, &age, hobby); !feof(input); fscanf(input, "%s %d %s\n", name, &age, hobby)) {    *// tmp variable to check this ahge is already included*  char isIncluded = 0;  *// until end of the ages array*  *for* (int i = 0; i < lines && ages[i] >= 0; i++) {  *// if found, update isIncluded*  *if* (ages[i] == age) {  isIncluded = 1;  }  }  *// add if it is not found.*  *if* (!isIncluded) {  ages[currentAgesIndex] = age;  currentAgesIndex++;  }  }  *// return to location specified*  fseek(input, loc, SEEK\_SET);  *return* ages;  }  *// group by ages*  int groupByAge(FILE \*input) {  *// go to beginning*  fseek(input, 0, SEEK\_SET);  *// read line and ages.*  int lines = howManyLines(input);  int \*ages = getAges(input);  *// open file "age.txt" to save*  FILE \*fp = (FILE \*)safeFileOpen("age.txt", "w");    *// for all lines*  *for* (int i = 0; i < lines && ages[i] >= 0; i++) {  *// temporary variable.*  int currentAge = ages[i];  char name[10] = "";  int age = 0;  char hobby[10] = "";  *// read all line.*  *for* (fscanf(input, "%s %d %s\n", name, &age, hobby); !feof(input); fscanf(input, "%s %d %s\n", name, &age, hobby)) {  *// if age is current Age, write*  *if* (age == currentAge) {  fprintf(fp, "%s %d %s\n", name, age, hobby);  }  }  *// goto beginning.*  fseek(input, 0, SEEK\_SET);  }  fclose(fp);  free(ages);  *return* 0;  }  *// max length for individual hobbies*  const int hobbiesLength = 10;  *// get Hobbies for*  char \*\*getHobbies(FILE \*input) {  *// save current location tmp.*  long loc = ftell(input);    *// go to beginning.*  fseek(input, 0, SEEK\_SET);  *// read lines*  int lines = howManyLines(input);  *// allocate memory for string pointer*  char \*\*hobbies = (char \*\*)safeMemoryAllocation(sizeof(char \*) \* lines);  *// initialize each index*  *for* (int i = 0; i < lines; i++) {  hobbies[i] = (char \*)safeMemoryAllocation(sizeof(char) \* hobbiesLength);  strcpy(hobbies[i], "");  }    *// tmp variable*  char name[10] = "";  int age = 0;  char hobby[10] = "";  *// tmp variable to current Hobbies index*  int currentHobbiesIndex = 0;  *// for each lines, read info*  *for* (fscanf(input, "%s %d %s\n", name, &age, hobby); !feof(input); fscanf(input, "%s %d %s\n", name, &age, hobby)) {    *// check it is included.*  char isIncluded = 0;  *// for all hobbies, check it is included in the hobbies array*  *for* (int i = 0; i < lines && hobbies[i][0] != 0; i++) {  *if* (!strcmp(hobbies[i], hobby)) {  isIncluded = 1;  }  }  *// if it is not included, add to list.*  *if* (!isIncluded) {  strcpy(hobbies[currentHobbiesIndex], hobby);  currentHobbiesIndex++;  }  }  *// return to previously saved location.*  fseek(input, loc, SEEK\_SET);  *return* hobbies;  }  *// function to group by hobby.*  int groupByHobby(FILE \*input) {  *// go to beginning.*  fseek(input, 0, SEEK\_SET);  *// read lines*  int lines = howManyLines(input);  *// read hobbies*  char \*\*hobbies = getHobbies(input);  *// rfile pointer*  FILE \*fp = (FILE \*)safeFileOpen("hobby.txt", "w");  *// for all hobbies*  *for* (int i = 0; i < lines && strcmp(hobbies[i], ""); i++) {    *// tmp variable to save current looping hobby.*  char \*currentHobby = hobbies[i];  *// tmp variable*  char name[10] = "";  int age = 0;  char hobby[10] = "";  *// read current line until end*  *for* (fscanf(input, "%s %d %s\n", name, &age, hobby); !feof(input); fscanf(input, "%s %d %s\n", name, &age, hobby)) {  *// check current hobby at hobbies*  *if* (!strcmp(hobby, currentHobby)) {  fprintf(fp, "%s %d %s\n", name, age, hobby);  }  }  *// go to beginning*  fseek(input, 0, SEEK\_SET);  }  *// close file*  fclose(fp);  *// free all hobbies*  freeAll((void \*\*)hobbies, lines);  *return* 0;  }  int main() {  *// open file*  FILE \*fp = safeFileOpen("personal.txt", "r");  *// ask user which you want to group*  printf("Group by which? (a for age, h for hobby) : ");  *// save command*  char cmd;  *// read char from cmd.*  cmd = fgetc(stdin);  *// flush stdin.*  fflush(stdin);  *// if user wants to classify via age*  *if* (cmd == 'a' || cmd == 'A') {  *return* groupByAge(fp);  *// if user wants to classify via hobby*  } *else* *if* (cmd == 'h' || cmd == 'H') {  *return* groupByHobby(fp);  *// wrong command.*  } *else* {  printf("Wrong command! Exiting....\n");  exit(0);  }  } |

## 

## **OUTPUT (Screen Shots)**

|  |  |
| --- | --- |
| **#1** | **Classify via Age** |
|  | |
| **#2** | **Classify via Hobby** |
|  | |

**//(필요한 만큼 더 생성하여 실행 결과를 잘 파악할 수 있도록 제시)**

**psh010209@gc.gachon.ac.kr, 010-7661-7035**

**Sanghee Park**

# **개발 과제 #2:**

|  |  |
| --- | --- |
|  | **Sample input and calculate candidate’s vote percentage.** |

**Requirements**

* **Step 1: Understand the problem - 간략한 핵심 요구 조건/작업 설명/예시 작성**

**Sample by input via file and calculate candidate’s vote percentage**

* **Step 2: Outline a solution – 간단한 logic설명**

**Create candidate list and vote list.**

**For every candidate occurrences, add 1 to specific candidate**

**Divide by candidate lists. And get percentage and show to output.**

* **Step 3: Form a program structure – 프로그램 전체 구조 요약**

**Create candidate list from file**

**Create vote list via candidate list**

**For all lines,**

**if the person casted vote to candidate,**

**add 1 to that candidate**

* **Step 4: write a pseudo code – 수도코드 작성**

**Read file.**

**Read all lines and find how many candidates occurred.**

**Give each candidate a unique index.**

**Create a vote array.**

**For all lines,**

**Find current row’s casted vote’s candidate.**

**Find the candidate index by name**

**If currentLine % sampleRate == 0**

**Add vote[candidate index] 1.**

**For all candidates:**

**Print candidateName + (Current candidate’s vote / sampledWholePersonLength) \* 100 %**

## **SOURCE CODE with comments**

|  |
| --- |
| *#include* <stdio.h>  *#include* <stdlib.h>  *#include* <string.h>  */\* common \*/*  *// open file, when it failed, exit the program*  FILE \*safeFileOpen(char \*fileName, char\* fileMode) {  FILE \*fp = fopen(fileName, fileMode);  *if* (fp == NULL) {  printf("Failed to open %s! Check the file permission or Disk S.M.A.R.T.!\n", fileName);  exit(1);  }  *return* fp;  }  *// malloc but when it failed, exit the program*  void \*safeMemoryAllocation(size\_t size) {  void \*memAddr = (void \*)malloc(size);  *if* (memAddr == NULL) {  printf("Failed to allocate memory! Check is it out of memory!");  exit(1);  }  *return* memAddr;  }  *// free all dynamically allocated double pointer*  void freeAll(void \*\*ptr, int length) {  *for* (int i = 0; i < length; i++) {  free(ptr[i]);  }  free(ptr);  }  */\* File Operations \*/*  *// Check how many line return characters are found in the file*  int howManyLines(FILE \*input) {  *// current location of file*  long a = ftell(input);  *// files have at least 1 lines*  int count = 1;  *// go to beginning*  fseek(input, 0, SEEK\_SET);  *// for all characters*  *for* (char c = 0; !feof(input); c=getc(input)) {  *// if line return increase 1*  *if* (c == '\n') {  count++;  }  }  *// to the previous location*  fseek(input, a, SEEK\_SET);  *return* count;  }  */\* internal stuff \*/*  *// maximum length of individual region*  const int regionsLength = 20;  *// maximum length of individual candidate.*  const int candidatesLength = 20;  *// get array of candidates*  char \*\*getCandidates(FILE \*input) {  *// get current location of file pointer*  long loc = ftell(input);  *// go to most front*  fseek(input, 0, SEEK\_SET);  *// check how many lines are found in the file*  int lines = howManyLines(input);  *// allocate memory for array*  char \*\*candidates = (char \*\*)safeMemoryAllocation(sizeof(char \*) \* lines);  *// initialize memory for imndividual candidates.*  *for* (int i = 0; i < lines; i++) {  candidates[i] = (char \*)safeMemoryAllocation(sizeof(char \*) \* candidatesLength);  strcpy(candidates[i], "");  }  *// temporary variables*  char regionName[regionsLength] = "";  int age = 0;  char candidateName[candidatesLength] = "";  *// current candidate index.*  int candidateIndex = 0;  *// for every indexes*  *for* (fscanf(input, "%s %d %s\n", regionName, &age, candidateName); !feof(input); fscanf(input, "%s %d %s\n", regionName, &age, candidateName)) {    *// tmp variable for save is this candidate was already included.*  char isIncluded = 0;  *// for all candidates in the list*  *for* (int i = 0; i < lines && candidates[i][0] != 0; i++) {  *// if candidate is found, change it is included*  *if* (!strcmp(candidates[i], candidateName)) {  isIncluded = 1;  }  }  *// if it is not included copy string.*  *if* (!isIncluded) {  strcpy(candidates[candidateIndex], candidateName);  candidateIndex++;  }  }  *// seek to prev. location*  fseek(input, loc, SEEK\_SET);  *return* candidates;  }  */\* main \*/*  int main() {  *// open file*  FILE \*fp = safeFileOpen("vote.txt", "r");  *// read line count*  int lineCount = howManyLines(fp);  *// read candidates*  char \*\*candidates = getCandidates(fp);  *// check how many candidates are in the candidates*  int candidatesCount = 0;  *// checking loop*  *for* (candidatesCount = 0; candidates[candidatesCount][0] != 0; candidatesCount++) { }  *// dynamically allocate for saving how many votes.*  int \*candidatesVotes = (int \*)safeMemoryAllocation(sizeof(int) \* candidatesCount);  *for* (int i = 0; i < candidatesCount; i++) {  candidatesVotes[i] = 0;  }  *// sample rate: ex. 2-4-6 -> sampleRate = 2*  *// ex. 3-6-9 -> sampleRate = 3*  int sampleRate = 1;  *// ask sampleRate*  printf("Enter the sampleRate for the vote : ");  scanf("%d", &sampleRate);  int totalSamples = 0;  *// for every lines,*  *for* (int i = 1; i <= lineCount && !feof(fp); i++) {  char regionName[regionsLength] = "";  int age = 0;  char candidateName[candidatesLength] = "";  *if* (feof(fp)) *break*;  fscanf(fp, "%s %d %s\n", regionName, &age, candidateName);  *// check it is current sample rate.*  *if* (i % sampleRate == 0) {  *// if it is, check for all candidates.*  *for* (int j = 0; j < candidatesCount; j++) {  *// write for cadidatesVotes.ddxsesesdsdsrasdfytrewaertyuysatytrd*  *if* (!strcmp(candidates[j], candidateName)) {  candidatesVotes[j]++;  totalSamples++;  }  }  }  }  *// for all candidates, print vote percents*  *for* (int i = 0; i < candidatesCount; i++) {  printf("%s: %.1lf%%\n", candidates[i], ((double)candidatesVotes[i] / totalSamples) \* 100);  }  *// free section.*  freeAll((void \*\*)candidates, lineCount);  free(candidatesVotes);  fclose(fp);  *return* 0;  } |

## **OUTPUT (Screen Shots)**

|  |  |
| --- | --- |
| **#1** | **sampleRate: 1 – 1 -> 2 -> 3** |
|  | |

|  |  |
| --- | --- |
| **#2** | **sampleRate: 2 – 2 -> 4 -> 6** |
|  | |

|  |  |
| --- | --- |
| **#3** | **sampleRate: 3 – 3 -> 6 -> 9** |
|  | |

|  |  |
| --- | --- |
| **#4** | **sampleRate: 10 – 10 -> 20 -> 30** |
|  | |