# **개발 과제 #1:**

|  |  |
| --- | --- |
|  | **Implement horizontal decomposition on provided file** |

**Requirements**

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

**Read from file to create table, decomposite table by horizontal index**

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

**Read File**

**Print to file if adequate row is found.**

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

**Read file,**

**Read age from line.**

**for each 10,19 / 20,29 / 30,39 / 40,49  
 for all lines  
 print to file if adequate  
 goto beginning of file**

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

**Read file**

**For each lines,**

**For i from 1 to 4 (both-inclusive)**

**If i\*10 <= age <= i\*10+9  
 print to file**

## **SOURCE CODE with comments**

|  |
| --- |
| *#include* <stdio.h>  *#include* <stdlib.h>  *#include* <string.h>  */\* common \*/*  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;  }  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;  }  int howManyLines(FILE \*input) {  long a = ftell(input);  int count = 1;  fseek(input, 0, SEEK\_SET);  *for* (char c = 0; !feof(input); c=getc(input)) {  *if* (c == '\n') {  count++;  }  }  fseek(input, a, SEEK\_SET);  *return* count;  }  */\* struct \*/*  struct PersonalInfo {  char name[10];  int age;  char hobby[10];  };  int main() {  FILE \*fp = safeFileOpen("personal.txt", "r");  int lineCount = howManyLines(fp);    struct PersonalInfo \*data = safeMemoryAllocation(sizeof(struct PersonalInfo) \* lineCount);  *for* (int i = 0; i < lineCount; i++) {  char name[10] = "";  int age = 0;  char hobby[10] = "";  fscanf(fp, "%s %d %s", name, &age, hobby);  strcpy(data[i].name, name);  data[i].age = age;  strcpy(data[i].hobby, hobby);  }  */\* main Decomposing algorithm \*/*  FILE \*out = safeFileOpen("output.txt", "w");    *for* (int i = 1; i <= 4; i++) {  fprintf(out, "Age from %d to %d\n", i\*10, i\*10+9);  fprintf(out, "--------------------\n");  *for* (int j = 0; j < lineCount; j++) {  *if* (i\*10 <= data[j].age && data[j].age <= i\*10+9) {  fprintf(out, "%s %d %s\n", data[j].name, data[j].age, data[j].hobby);  }  }  fprintf(out, "\n");  }  fclose(out);  fclose(fp);  free(data);  } |

## **OUTPUT (Screen Shots)**

|  |  |
| --- | --- |
| **#1** | **Classification Example** |
| **모니터, 스크린샷, 검은색이(가) 표시된 사진  자동 생성된 설명** | |

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

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

**Sanghee Park**

# **개발 과제 #2:**

|  |  |
| --- | --- |
|  | **Implement vertical decomposition on provided file** |

**Requirements**

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

**Read from file to create table, decomposite table by selected vertical column**

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

**Read File  
For each row: print name and selected column**

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

**Read file,**

**Ask which column you are selecting.**

**For each row: print name and selected column**

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

**Read file**

**Ask user which column you are selecting**

**Save it to temporary variable for later use**

**For each lines, print name and selected column via saved temporary variable**

## **SOURCE CODE with comments**

|  |
| --- |
| *#include* <stdio.h>  *#include* <stdlib.h>  *#include* <string.h>  */\* common \*/*  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;  }  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;  }  int howManyLines(FILE \*input) {  long a = ftell(input);  int count = 1;  fseek(input, 0, SEEK\_SET);  *for* (char c = 0; !feof(input); c=getc(input)) {  *if* (c == '\n') {  count++;  }  }  fseek(input, a, SEEK\_SET);  *return* count;  }  */\* struct \*/*  struct PersonalInfo {  char name[10];  int age;  char hobby[10];  };  int main() {  FILE \*fp = safeFileOpen("personal.txt", "r");  int lineCount = howManyLines(fp);    struct PersonalInfo \*data = safeMemoryAllocation(sizeof(struct PersonalInfo) \* lineCount);  *for* (int i = 0; i < lineCount; i++) {  char name[10] = "";  int age = 0;  char hobby[10] = "";  fscanf(fp, "%s %d %s", name, &age, hobby);  strcpy(data[i].name, name);  data[i].age = age;  strcpy(data[i].hobby, hobby);  }  char cmd = 0;  printf("Which data you want to vertical decomp? (a: age, h: hobby) : ");  scanf("%c", &cmd);  fflush(stdin);  *if* (cmd == 'a' || cmd == 'A') {  printf("decomp-ing via age.\n");  } *else* *if* (cmd == 'h' || cmd == 'H') {  printf("decomp-ing via hobby.\n");  } *else* {  printf("invalid command. exiting...\n");  *return* 0;  }  */\* main Decomposing algorithm \*/*  FILE \*out = safeFileOpen("output.txt", "w");    *for* (int j = 0; j < lineCount; j++) {  *if* (cmd == 'a' || cmd == 'A') {  fprintf(out, "%s %d\n", data[j].name, data[j].age);  } *else* *if* (cmd == 'h' || cmd == 'H') {  fprintf(out, "%s %s\n", data[j].name, data[j].hobby);  }  }  fclose(out);  fclose(fp);  free(data);  } |

## **OUTPUT (Screen Shots)**

|  |  |
| --- | --- |
| **#1** | **Classification Example** |
|  | |

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

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

**Sanghee Park**

# **개발 과제 #3:**

|  |  |
| --- | --- |
|  | **Update salary with the version control via filename** |

**Requirements**

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

**Update salary file data with Version control via filename**

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

**Read previous file**

**Manipulate Data**

**Write new file with new version name**

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

**Read file, Read to memory.**

**Run Manipulation Job for ver2**

**Write to ver2 file. Clear memory.**

**Read ver2 file, Read to memory**

**Run Manipulation Job for ver3**

**Write to ver3 file. Clear memory**

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

**Read file**

**Allocate memory and save file data to memory**

**If 40<= age <= 49, Multiply by 1.1**

**Write ver2 file**

**Free memory**

**Read ver2 file**

**Allocate memory and save file data to memory**

**If 30<= age <= 39, Multiply by 1.2**

**Free memory**

## **SOURCE CODE with comments**

|  |
| --- |
| *#include* <stdio.h>  *#include* <stdlib.h>  *#include* <string.h>  */\* common \*/*  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;  }  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;  }  int howManyLines(FILE \*input) {  long a = ftell(input);  int count = 1;  fseek(input, 0, SEEK\_SET);  *for* (char c = 0; !feof(input); c=getc(input)) {  *if* (c == '\n') {  count++;  }  }  fseek(input, a, SEEK\_SET);  *return* count;  }  */\* struct \*/*  struct PersonalInfo {  char name[10];  int age;  float salary;  };  int main() {  FILE \*fp = safeFileOpen("salary\_v1.txt", "r");  int lineCount = howManyLines(fp);    struct PersonalInfo \*data = safeMemoryAllocation(sizeof(struct PersonalInfo) \* lineCount);  *for* (int i = 0; i < lineCount; i++) {  char name[10] = "";  int age = 0;  float salary = 0.0;  fscanf(fp, "%s %d %f", name, &age, &salary);  strcpy(data[i].name, name);  data[i].age = age;  data[i].salary = salary;  }  fclose(fp);  */\* main Decomposing algorithm \*/*  FILE \*v2 = safeFileOpen("salary\_v2.txt", "w");    *for* (int j = 0; j < lineCount; j++) {  *if* (40 <= data[j].age && data[j].age <= 49) {  data[j].salary \*= 1.1;  }  fprintf(v2, "%s %d %.1f\n", data[j].name, data[j].age, data[j].salary);  }  fclose(v2);  free(data);  v2 = safeFileOpen("salary\_v2.txt", "r");  lineCount = howManyLines(v2);  data = safeMemoryAllocation(sizeof(struct PersonalInfo) \* lineCount);  *for* (int i = 0; i < lineCount; i++) {  char name[10] = "";  int age = 0;  float salary = 0.0;  fscanf(fp, "%s %d %f", name, &age, &salary);  strcpy(data[i].name, name);  data[i].age = age;  data[i].salary = salary;  }  fclose(v2);  */\* main Decomposing algorithm \*/*  FILE \*v3 = safeFileOpen("salary\_v3.txt", "w");  *for* (int j = 0; j < lineCount; j++) {  *if* (30 <= data[j].age && data[j].age <= 39) {  data[j].salary \*= 1.2;  }  fprintf(v3, "%s %d %.1f\n", data[j].name, data[j].age, data[j].salary);  }  fclose(v3);  free(data);  } |

## **OUTPUT (Screen Shots)**

|  |  |
| --- | --- |
| **#1** | **Manipulation via version** |
|  | |