C프로그래밍및실습

사원관리프로그램

진척 보고서 #1

제출일자: 2023.11.26
제출자명: 정지우
제출자학번: 18115

1. 프로젝트 목표

1) 배경 및 필요성

대기업이나 중견기업의 경우 값비싼 사원관리 프로그램을 구매하여 사용하지만 중소기업이나 스타트업의 경우 자본금이 부족하기에 이런 사원관리 프로그램까지 구매해서 사용할 여력이 없거나 빠듯한 경우가 많다. 또한 지나치게 복잡하고 많은 기능들을 포함하고 있어 이로인한 사용에 애로사항이 발생하기도 한다. 이 문제를 해결하기 위해 C언어로도 손쉽게 만들 수 있는 사원들의 목록을 관리하고 추가 및 수정이 가능하며 급여, 입사날짜, 직책 등을 저장해서 리스트화 해주는 프로그램이 필요하다.

2) 프로젝트 목표

회사 직원들의 직책, 급여, 입사날짜 등의 기본 정보와 사원들의 휴가 등을 관리하는 사원관리 프로그램을 만드는 것을 목표로 한다.

3) 차별점

기존 사원관리 프로그램의 경우 값비싼 것은 둘째치고 개인이 따라만들기에는 매우 복잡하고 어려운 경우가 많았으며 필요없는 기능도 다수 포함되어있어 오히려 필요한 기능만 골라쓰기 힘든 경우가 많았다. 하지만 이 프로젝트로 만드는 프로그램은 누구나 C언어로 따라 만들 수 있으며 실제 사무현장에서 꼭 필요한 기능들만 포함하고 있다. 또한 불필요한 것들 없이 담백하게 필요한 정보만을 출력한다.

2. 기능 계획

1) 기능 1 : 사원 정보 관리 기능

- 설명 :현재 회사에 재직중인 사원들의 정보를 입력하는 기능이다.

(1) 세부 기능 1 : 사원 추가 시스템

- 설명 : 직책,급여,입사날짜,나이,거주지역 등을 사원별로 입력하는 기능이다.

(2)세부 기능 2: 목록 수정 기능

- 설명 : 입력된 사원 중에서 원하는 사원의 원하는 카테고리를 수정하는 기능이다. 수정을 하게되면 이전 기록은 사라지고 수정한 기록만 남게 된다.

(3)세부 기능 3: 목록 출력 기능

-설명: 사원 테이블에 입력된 사원 정보를 사원별로 모두 출력해준다.

2) 기능 2 : 퇴사자 관리 기능

- 설명 : 퇴사한 직원을 따로 목록으로 만들어 관리하는 기능이다. 직원의 퇴사를 입력하게 되면 퇴사자 명단으로 넘어가서 저장되며 퇴사자 명단을 한번에 출력해서 볼 수 있다.

2) 기능 3 : 휴가(연차) 관리 기능

(1)세부 기능 1: 휴가자 명단 관리 기능

- 설명 : 휴가를 간 직원들의 휴가 간 날짜, 휴가에서 돌아오는 날짜, 사유 등을 기록한 리스트를 보여주는 기능이다.

(2)세부 기능 2: 직원들의 연차 관리 기능

-설명 : 직원들의 남은 연차를 리스트화 해서 보여주고 수정할 수 있는 기능이다.

3. 진척사항

1) 기능 구현

(1) 사원 정보 입력 기능

[AddEmployee 함수]

<입출력>

1)입력

-사원의 이름을 구조체 배열 EmployeeTable에서 num_people번째에 employ_name 문자열에 입력 받는다.

-사원의 입사날짜를 구조체 배열 EmployeeTable에서 num_people번째에 start_company 문자열에 입력 받는다.

-사원의 직급을 구조체 배열 EmployeeTable에서 num_people번째에 role 문자열에 입력 받는다.

-사원의 나이를 구조체 배열 EmployeeTable에서 num_people번째에 age변수에 입력 받는다.

-사원의 거주지를 구조체 배열 EmployeeTable에서 num_people번째에 residence 문자열에 입력 받는다.

-사원이 한명 씩 입력 될 때 마다 num_people의 수가 증가한다.

2)출력

-출력 값은 존재하지 않는다

<설명>

추가할 사원의 나이,거주지,직급,입사날짜 등의 정보를 한명씩 입력하여 EmployeeTable이라는 구조체에 저장하여 사원의 정보가 담긴 목록을 만든다. 이 과정에서 입력된 사원의 수가 할당된 용량 capacity에 도달하면 ResizeTable이라는 사용자 정의 함수를 통해 새로운 크기의 메모리를 할당하고 기존 데이터를 새 메모리에 복사한다.

<적용된 배운 내용>

-구조체, 입출력함수, 배열, 함수, 포인터, 동적메모리

<코드 스크린샷>

```
void AddEmployee(struct EmployData *table, int *num_people) {
  ResizeTable();

  printf("사원 이름을 입력하세요: ");
  scanf_s("%s", table[*num_people].employ_name,(int)sizeof(table[*num_people].employ_name));
  printf("사원의 입사날짜를 입력하세요(ex:1900/01/01): ");
  scanf_s("%s", table[*num_people].start_company,(int)sizeof(table[*num_people].start_company));
  printf("사원의 거주지역을 입력하세요: ");
  scanf_s("%s", table[*num_people].residence,(int)sizeof(table[*num_people].residence));
  printf("사원의 직급을 입력하세요: ");
  scanf_s("%s", table[*num_people].role,(int)sizeof(table[*num_people].role));
  printf("사원의 나이를 입력하세요: ");
  scanf_s("%d", &table[*num_people].age);

  (*num_people)++;
}
```

[ResizeTable 함수]

<입출력>

1)입력

-입력값은 존재하지 않는다.

2)출력

-출력값도 존재하지 않는다.

<설명>

사원 정보를 입력하는 과정에서 ResizeTable이라는 사용자 정의 함수를 통해 입력된 사원의 수가 할당된 용량 capacity에 도달하면 realloc함수를 이용해 새로운 크기의 메모리를 할당한다.

<적용된 배운 내용>

-출력함수, 함수, 동적메모리, 조건문

<코드 스크린샷>

```
void ResizeTable() {
  if (num_people == capacity) {
    capacity = (capacity == 0) ? INITIAL_CAPACITY : capacity * 2;
    EmployeeTable = realloc(EmployeeTable, capacity * sizeof(struct EmployData));
    if (EmployeeTable == NULL) {
      printf("메모리 할당 실패₩n");
      exit(EXIT_FAILURE);
    }
  }
}
```

(2) 사원 정보 수정 기능

[FindPerson 함수]

<입출력>

1)입력

-struct EmployData *table: 사원의 정보를 담고 있는 구조체 배열에 대한 포인터를 입력 받는다.

-num_people: 배열에 현재 저장된 사원 정보의 수를 입력 받는다.

-char *name: 찾고자 하는 사원의 이름이 담긴 문자열의 포인터를 입력 받는다.

2)출력

-찾고자하는 사원의 이름을 사원의 정보를 담고 있는 구조체 배열에서 찾으면 몇 번째에 위치했는지 그 i번째 값을 출력한다.

-찾고자하는 사원의 이름이 사원의 정보를 담고 있는 구조체 배열에 존재하지 않으면 무조건 -1을 출력한다.

<설명>

수정하고자 하는 사원의 이름을 입력하면 사원테이블에서 해당 사원을 찾아서 해당사원의 인덱스를 반환해준다. 해당 사원이 존재하지 않을 경우 잘못된 이름이 입력되었다는 메시지가 출력된다.

<적용된 배운 내용>

-조건문(switch), 구조체, 입출력함수, 배열, 함수, 포인터, 동적메모리

<코드 스크린샷>

```
int FindPerson(const struct EmployData *table, int num_people,
               const char *name) {
  for (int i = 0; i < num_people; ++i) {
    if (strcmp(table[i].employ_name, name) == 0) {
      return i;
    }
  }
  return -1;  // 찾지 못한 경우 -1 반환
}
```

[UpdateEmployee 함수]

<입출력>

1)입력

-struct EmployData *table: 사원의 정보를 가지고 있는 구조체 배열의 포인터를 입력 받는다. 이 배열에서 FindPerson함수를 통해 정보를 변경하고자 하는 사원을 찾는다

- num_people: 배열에 현재 저장된 사원 정보의 수를 입력 받는다.

-update_choice: 제시된 4개의 수정 가능 항목 중 수정할 항목의 번호를 입력받는다.

-수정할 항목을 고른 후 수정값을 입력하면 각각 항목에 맞게 사원 구조체 배열의 age,startcompany,role,residence에 수정값이 입력된다.

-editing: 초기 값은 1로 되어있어서 while문을 무한 반복하게 해주며 종료 값을 입력하면 0으로 바뀌고 while반복문을 종료한다.

2)출력

-출력값은 존재하지 않는다

<설명>

수정하고자 하는 사원 이름을 알맞게 입력했으면 수정할 항목을 고르는 메시지가 출력되고 수정을 원하는 항목을 골라서 수정 값을 입력하면 해당 사원의 정보가 수정이 된다. 이때 종료를 의미하는 5를 입력하기 전까지 원하는 만큼 사원의 정보를 수정할 수 있다.

<적용된 배운 내용>

-조건문(switch), 구조체, 입출력함수, 배열, 함수, 포인터, 동적메모리

<코드 스크린샷>

```
void UpdateEmployee(struct EmployData *table, int num_people) {
  char TargetName[50];
  printf("₩n수정할 사람의 이름을 입력하세요: ");
  scanf_s("%s", TargetName, (int)sizeof(TargetName));

  int peopleIndex = FindPerson(table, num_people, TargetName);

  if (peopleIndex == -1) {
    printf("잘못된 이름을 입력했습니다.₩n");
    return;  // 처음 메뉴 선택으로 돌아감
  }
  int editing = 1;
  while (editing) {
    int update_choice;
    printf("-----------------------------₩n");
    printf("수정할 항목을 선택하세요:₩n");
    printf("1. 나이₩n2. 거주지₩n3. 직급₩n4. 근무 연도₩n5. 종료₩n");
    printf("-----------------------------₩n");
    printf("메뉴: ");
    scanf_s("%d", &update_choice);

    switch (update_choice) {
      case 1:
        printf("새로운 나이 입력: ");
        scanf_s("%d", &table[peopleIndex].age);
        break;
      case 2:
        printf("새로운 거주지 입력: ");
        scanf_s("%s", table[peopleIndex].residence,(int)sizeof(table[peopleIndex].residence));
        break;
      case 3:
        printf("새로운 직급 입력: ");
        scanf_s("%s", table[peopleIndex].role,(int)sizeof(table[peopleIndex].role));
        break;
      case 4:
        printf("새로운 근무 연도 입력: ");
        scanf_s("%s", table[peopleIndex].start_company,(int)sizeof(table[peopleIndex].start_company));
        break;
      case 5:
        printf("수정된 사원 정보:₩n");
        printf("이름: %s₩n", table[peopleIndex].employ_name);
        printf("나이: %d₩n", table[peopleIndex].age);
        printf("입사 날짜: %s₩n", table[peopleIndex].start_company);
        printf("직급: %s₩n", table[peopleIndex].role);
        printf("거주지: %s₩n", table[peopleIndex].residence);
        editing = 0;
        break;
      default:
        printf("잘못된 선택입니다.₩n");
    }
  }
}
```

(3) 사원 정보 출력 기능

[PrintEmployee 함수]

<입출력>

1)입력

-struct EmployData *table: 사원의 정보를 담고 있는 구조체 배열에 대한 포인터를 입력 받는다.

- num_people: 배열에 현재 저장된 사원 정보의 수를 입력 받는다.

2)출력

-반환값은 존재하지 않으며 사원테이블에 입력된 사원별로 사원 정보를 모두 출력해준다.

<설명>

입력된 나이, 직급, 거주지, 입사날짜 등의 정보를 사원별로 모든 사원에 대해 출력해준다.

<적용된 배운 내용>

-구조체, 출력함수, 배열, 함수, 포인터

<코드 스크린샷>

```
void PrintEmployee(const struct EmployData *table, int num_people) {
  for (int i = 0; i < num_people; i++) {
    printf("------------------------------₩n");
    printf("이름: %s₩n", table[i].employ_name);
    printf("나이: %d₩n", table[i].age);
    printf("입사 날짜: %s₩n", table[i].start_company);
    printf("직급: %s₩n", table[i].role);
    printf("거주지: %s₩n", table[i].residence);
    printf("------------------------------₩n");
  }
}
```

2) 테스트 결과

(1) 사원 정보 입력 기능

- 설명

처음에 원하는 메뉴를 입력하는 문구가 출력되면 사원 명단 신규 추가인 1번을 입력하면 사원을 입력하는 문구가 출력된다. 차례차례 이름부터 입사날짜,거주지,직급,나이를 입력하는 문구가 출력되면 각각 정보를 입력하면 사원 정보 입력이 완료되고 다시 처음에 메뉴를 선택으로 돌아간다..

- 테스트 결과 스크린샷

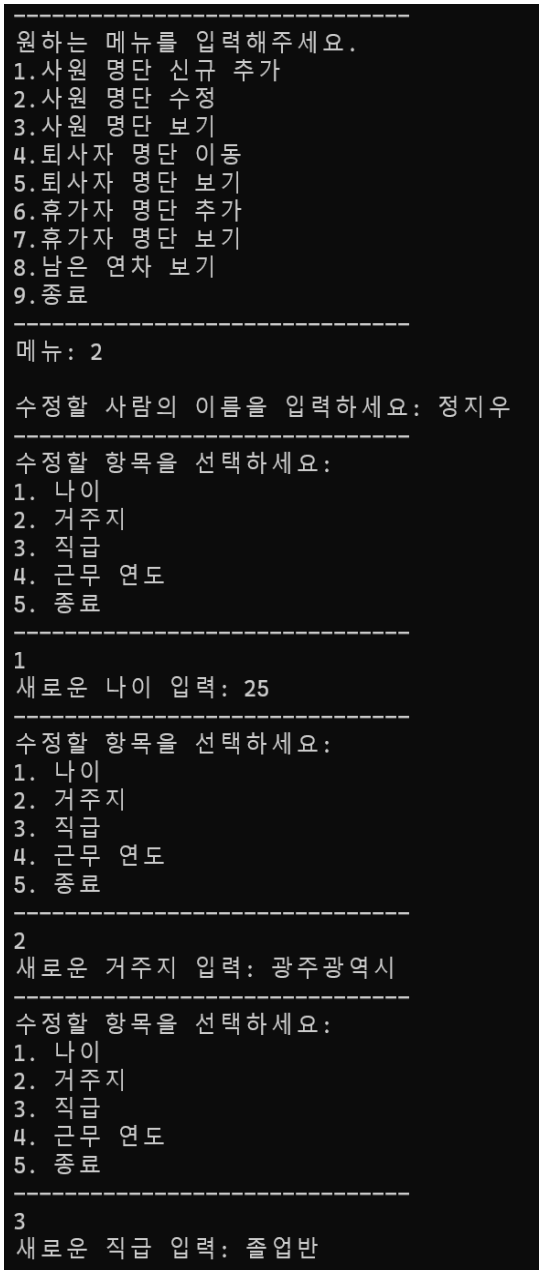
```
------------------------------
원하는 메뉴를 입력해주세요.
1.사원 명단 신규 추가
2.사원 명단 수정
3.사원 명단 보기
4.퇴사자 명단 이동
5.퇴사자 명단 보기
6.휴가자 명단 추가
7.휴가자 명단 보기
8.남은 연차 보기
9.종료
------------------------------
메뉴: 1
사원 이름을 입력하세요: 정지우
사원의 입사날짜를 입력하세요(ex:1900/01/01): 2018/03/02
사원의 거주지역을 입력하세요: 광주
사원의 직급을 입력하세요: 4학년
사원의 나이를 입력하세요: 24
------------------------------
원하는 메뉴를 입력해주세요.
1.사원 명단 신규 추가
2.사원 명단 수정
3.사원 명단 보기
4.퇴사자 명단 이동
5.퇴사자 명단 보기
6.휴가자 명단 추가
7.휴가자 명단 보기
8.남은 연차 보기
9.종료
------------------------------
메뉴:
```

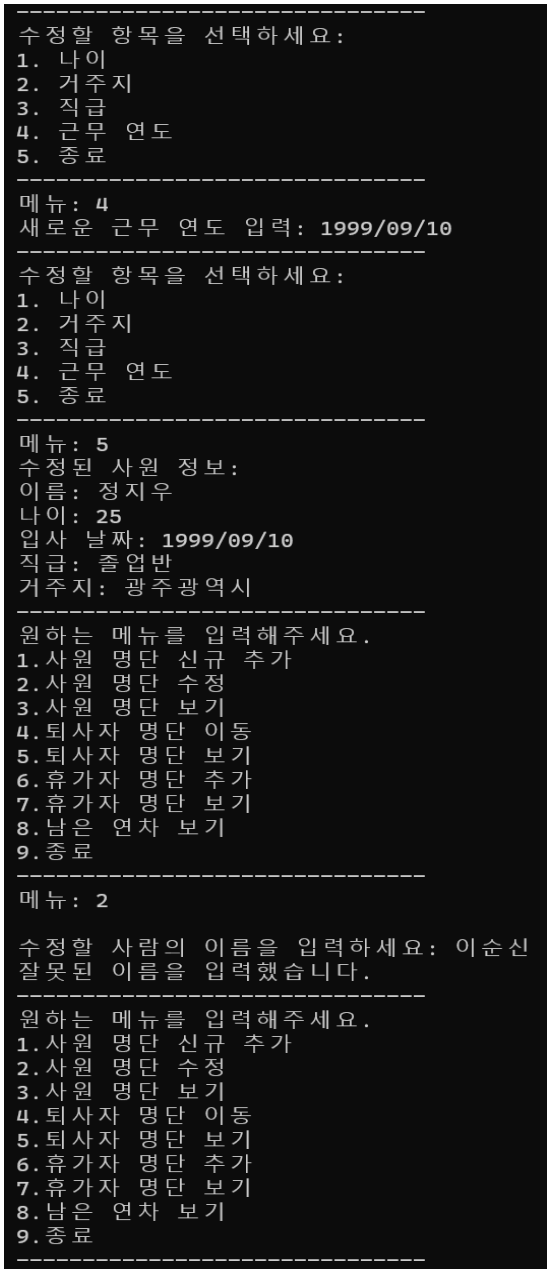
(2) 사원 정보 수정 기능

- 설명

사원 정보를 수정하고자 하는 사원의 이름을 입력하면 수정할 항목들을 고르는 문구가 출력된다. 수정할 해당 항목의 번호를 입력하면 수정할 정보를 입력할 수 있으며 수정이 다 끝나고 종료 5를 입력하면 수정이 종료되고 입력한 수정 정보가 출력되고 처음 메뉴를 고르는 화면으로 돌아간다.

- 테스트 결과 스크린샷



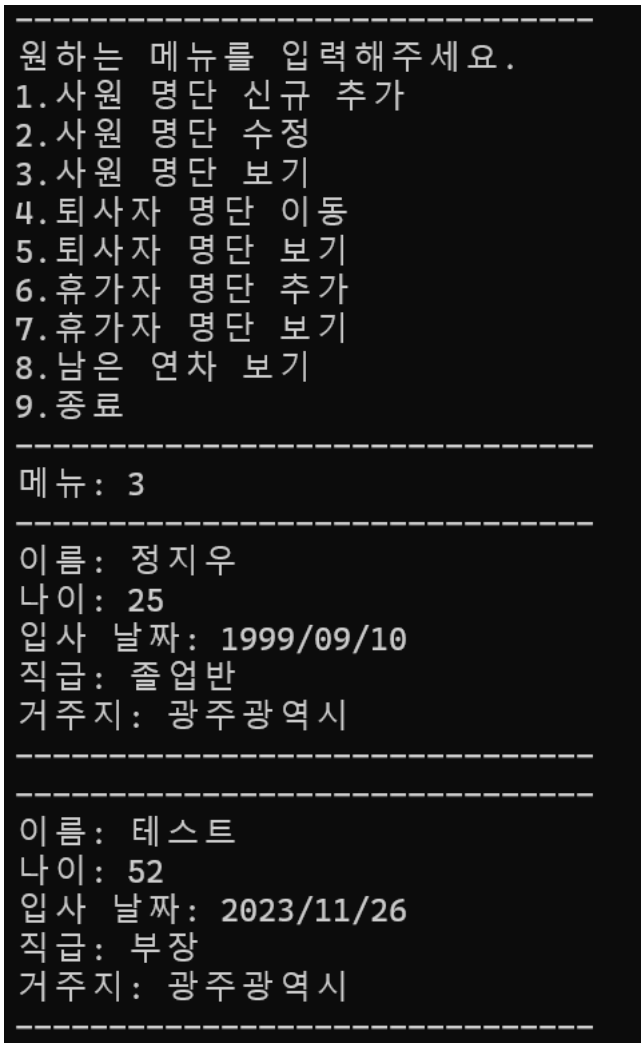


(3) 사원 정보 출력 기능

- 설명

입력된 사원 정보를 보고싶다면 초기 원하는 메뉴를 고르는 문구가 출력되면 사원 명단 보기를 의미하는 3을 입력한다. 3을 입력하면 현재 저장된 모든 사원의 정보가 사원별로 출력되어 확인할 수 있다.

- 테스트 결과 스크린샷



4. 계획 대비 변경 사항

1) 사원 정보 입력 기능 일부 수정

- 이전

초기에는 사원정보 구조체에 최대 용량을 100으로 지정해기에 이미 정해진 크기의 배열의 메모리에 입력하는 사원의 정보를 저장해서 채워나가는 방식이었다.

- 이후

사원정보 구조체에 메모리를 지정하지 않고 동적할당을 통해 새로운 사원의 정보가 입력될때 마다 메모리를 증가시켜 저장하도록 변경하였다.

- 사유

미리 메모리를 할당하는 것보다 상황에 따라 원하는 크기만큼 메모리를 할당하는 동적할당을 이용하는 것이 훨씬 메모리 낭비가 없고 효율적이고 유연하다고 판단하였다.

2) 사원 정보 출력 기능 추가

- 이전

사원 구조체 배열에 사원 정보를 입력하고 수정하는 기능만 있었지 출력하는 기능이 존재히지않아 입력한 사원 정보를 볼 수 없었다.

- 이후

사원 정보를 모두 출력하는 기능을 통해 내가 입력한 사원 정보를 사원별로 모두 출력하여 확인할 수 있다.

- 사유

제안서를 쓸 때 프로그램 기능을 구상하는 과정에서 사원 명단을 출력하는 기능을 누락되었다. 따라서 이번 진척보고서를 쓰는 과정에서 기능 1의 세부기능에 출력하는 기능도 구현하여 추가하였다.

5. 프로젝트 일정

업무		11/3	11/6	11/24	12/10	12/23
제안서 작성		완료				
기능1	세부기능1		완료			
	세부기능2		완료			
	세부기능3		완료			
기능2				진행중		
기능3	세부기능1				---------->	
	세부기능2				---------->	

기능1:사원 정보 관리 기능

->세부기능1:사원정보 입력

->세부기능1:사원정보 수정

->세부기능1:사원정보 출력

기능2:퇴사자 관리 기능

기능3:휴가(연차)관리 기능

->휴가자 명단 관리

->사원 연차 관리