C – 은행 시스템 프로젝트

BankSystem

은행 업무 프로젝트 개요

◆ 은행 업무 프로젝트

은행 계좌 구조체를 만들고, 은행 업무 기능 만들기

- 은행 업무 프로젝트 단계

step1. 문제 정의하기

step2. 구조체 정의하고 관계도 그리기

step3. 은행 업무 기능 설계하고 구현하기

step4. 파일 배포(서비스)

step1. 문제 정의하기

◆ 프로그램 시나리오

- 계정(BankAccount) 구조체는 계좌 번호, 계좌주, 잔액으로 구성되어 있다.
- BankAccount 배열을 100개 생성한다.
- main 함수에는 계좌 생성, 입금, 출금, 계좌 목록, 종료 등의 메뉴가 있다.

계좌 번호	계좌주	금액
100-200-3000	한강	1000
101-200-3000	이이슬	2000
102-200-3000	알파고	3000

step1. 문제 정의하기

◆ 메뉴별 결과 화면

```
=====================================================
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====================================================
선택> 1
계좌 번호: 100-200-3000
계좌주: 한이슬
결과: 계좌가 생성되었습니다. (계좌 번호: 100-200-3000)
```

```
=====================================================
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====================================================
선택> 2
입금할 계좌번호(예: 000-000-0000): 100-200-3000
입금액: 20000
정상 처리 되었습니다. 현재 잔액: 20000
```

```
=====================================================
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====================================================
선택> 3
출금할 계좌번호(예: 000-000-0000): 100-200-3000
출금액: 10000
정상 처리 되었습니다. 현재 잔액: 10000
```

step1. 문제 정의하기

◆ 메뉴별 결과 화면

```
=====================================================
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====================================================
선택> 4
********** 계      좌      목      록 **********
계좌 번호: 100-200-3000, 계좌주: 한이슬, 잔액: 10000
```

@ 출금시 잔액 부족 오류

```
=====================================================
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====================================================
선택> 3
출금할 계좌번호(예: 000-000-0000): 100-200-3000
출금액: 50000
잔액이 부족하거나 올바른 금액이 아닙니다. 현재 잔액: 20000
```

@ 계좌 중복 오류

```
=====================================================
1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료
=====================================================
선택> 1
계좌 번호: 100-200-3000
이미 등록된 계좌입니다. 다시 입력하세요.
```

step2. 구조체 다이어그램

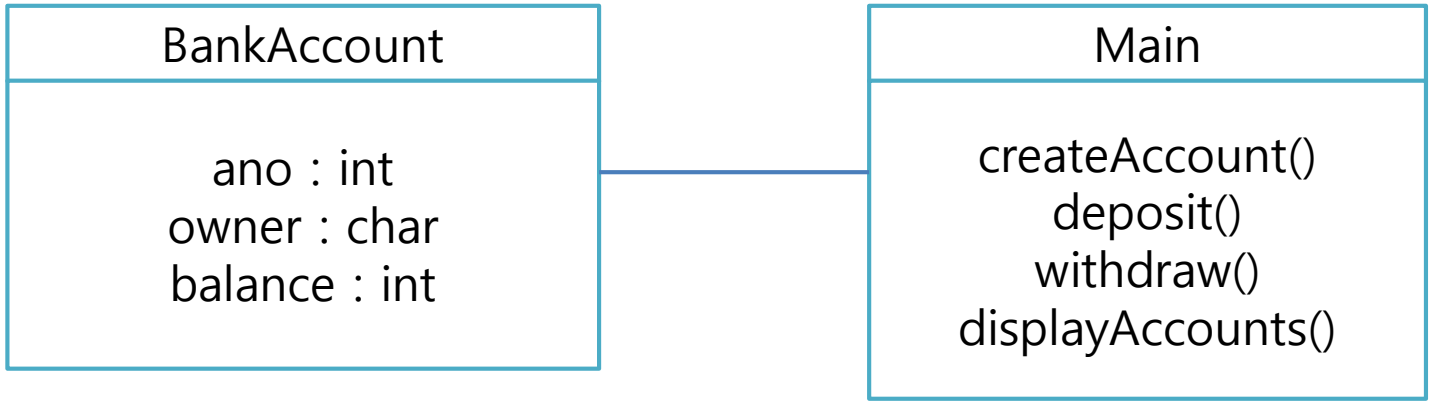
◆ **구조체 관계도 그리기**

BankAccount 구조체

계좌 번호
계좌주
잔액

main 함수

계좌 생성
입금
출금
계좌 목록



step2. 구조체 정의하기

- **BankAccount 구조체**

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <string.h> //strcpy()
#include <stdbool.h> //true/false 사용

#define MAX_ACCOUNTS 100    //최대 계정수
#define ANO_LEN 20          //계좌번호 크기
#define OWNER_LEN 30        //예금주 크기

typedef struct {
    char ano[ANO_LEN];      //계좌번호
    char owner[OWNER_LEN]; //예금주
    int balance;            //잔고
}BankAccount;

//전역 공간
BankAccount accounts[MAX_ACCOUNTS]; //계좌 배열 생성
int accountCount = 0; //현재 계좌 수
```

step3. 은행 업무 기능 설계, 구현

- **main() 함수**

```
int main()
{
    bool run = true;  //실행,종료
    int choice; //메뉴

    while (run)
    {
        printf("=====================================================\n");
        printf("1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료\n");
        printf("=====================================================\n");
        printf("선택> ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            createAccount();
            break;
        case 2:
            deposit();
            break;
```

step3. 은행 업무 기능 설계, 구현

- **main() 함수**

```
            case 3:
                withdraw();
                break;
            case 4:
                displayAccounts();
                break;
            case 5:
                printf("프로그램을 종료합니다.\n");
                run = false;
                break;  //정상 종료
            default:
                printf("잘못된 입력입니다. 다시 선택하세요.\n");
            }
        }

        system("pause");  //콘솔창 닫힘 방지(파일 배포시)

        return 0;
}
```

step3. 은행 업무 기능 설계, 구현

- **계좌 생성**

```
void createAccount() {
    char accountNumber[ANO_LEN];  //입력(계좌번호)

    if (accountCount >= MAX_ACCOUNTS) {
        printf("더 이상 계좌를 생성할 수 없습니다.\n");
        return;
    }

    printf("계좌 번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    //중복 검사
    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            printf("이미 등록된 계좌입니다. 다시 입력하세요.\n");
            return;
        }
    }
```

step3. 은행 업무 기능 설계, 구현

- **계좌 생성**

```
    //중복 없을때 계좌 생성
    strcpy(accounts[accountCount].ano, accountNumber); //계좌번호 복사
    printf("계좌주: ");
    scanf("%s", accounts[accountCount].owner);
    accounts[accountCount].balance = 0; //잔고

    printf("결과: 계좌가 생성되었습니다. (계좌 번호: %s)\n",
        accounts[accountCount].ano);

    accountCount++;  //다음 인덱스로 증가
}
```

step3. 은행 업무 기능 설계, 구현

- **계좌 목록**

```
void displayAccounts() {
    printf("********** 계     좌     목     록 **********\n");

    if (accountCount == 0)
    {
        printf("등록된 계좌가 없습니다.\n");
        return;
    }

    for (int i = 0; i < accountCount; i++)
    {
        printf("계좌 번호: %s, 계좌주: %s, 잔액: %d\n",
            accounts[i].ano, accounts[i].owner, accounts[i].balance);
    }
}
```

step3. 은행 업무 기능 설계, 구현

- 예금

```
void deposit() {
    char accountNumber[ANO_LEN];  //외부 입력(계좌번호)
    int amount;    //입금액
    bool found = false; //상태(계좌 찾음/못찾음)

    printf("입금할 계좌번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            found = true; //계좌 찾음

            printf("입금액: ");
            scanf("%d", &amount);
            if (amount < 0) {
                printf("올바른 금액이 아닙니다.\n");
                return; //즉시 종료
            }
```

step3. 은행 업무 기능 설계, 구현

- 예금

```
                else {
                    accounts[i].balance += amount;
                    printf("정상 처리 되었습니다. 현재 잔액: %d\n",
                                                    accounts[i].balance);
                    return;  //for문 탈출
                }
            }
        }

        if (!found) {
            printf("계좌를 찾을 수 없습니다.\n");
        }
}
```

step3. 은행 업무 기능 설계, 구현

- **출금**

```
void withdraw() {
    char accountNumber[ANO_LEN];  //외부 입력(계좌번호)
    int amount;  //출금액
    bool found = false; //상태(계좌 찾음/못찾음)

    printf("출금할 계좌번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            found = true; //계좌 찾음

            printf("출금액: ");
            scanf("%d", &amount);
            if (amount > accounts[i].balance || amount < 0) {
                printf("잔액이 부족하거나 올바른 금액이 아닙니다. "
                                "현재 잔액: %d\n", accounts[i].balance);
                return;
            }
```

step3. 은행 업무 기능 설계, 구현

- 출금

```
            else {
                accounts[i].balance -= amount;
                printf("정상 처리 되었습니다. 현재 잔액: %d\n",
                                        accounts[i].balance);
                return;  //for문 탈출
            }
        }
    }

    if (!found) {
        printf("계좌를 찾을 수 없습니다.\n");
    }
}
```

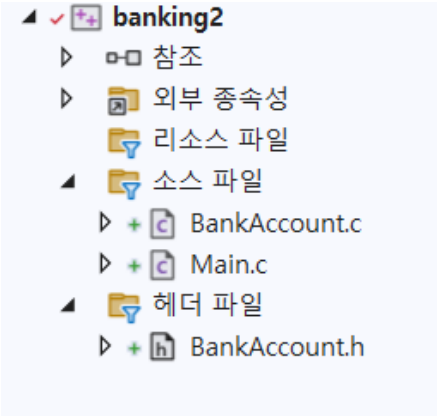
헤더 파일 사용하기

❖ 헤더파일 사용하기

- 다른 소스 파일에서 함수 또는 변수를 사용하는 방법이다.
- 헤더파일에서는 함수의 프로토 타입을 선언한다.
- 헤더파일 > 추가 > 새항목 > Calculator.h

❖ BankSystem 프로젝트 만들기

- BankAccount.h – 헤더 파일(구조체, 전역 변수, 함수 선언부)
- BankAccount.c – 구현 파일(변수 초기화, 함수 구현)
- Main.c – 실행 파일



헤더 파일로 분리하기

- **BankAccount.h**

```
#ifndef BANK_ACCOUNT_H  //조건부 컴파일 블럭
#define BANK_ACCOUNT_H  //매크로 이름(구조체 이름 중복 불가)

#define MAX_ACCOUNTS 100    //최대 계정수
#define ANO_LEN 20          //계좌번호 크기
#define OWNER_LEN 30        //예금주 크기

typedef struct {
    char ano[ANO_LEN];      //계좌번호
    char owner[OWNER_LEN]; //예금주
    int balance;            //잔고
}BankAccount;
```

헤더 파일로 분리하기

- **BankAccount.h**

```
//전역 변수 선언
//extern은 헤더 파일을 포함한 BankAccout.c에서만 정의할 수 있음
extern BankAccount accounts[MAX_ACCOUNTS]; //계좌 배열 생성
extern int accountCount; //현재 계좌 수

//계좌 관련 함수 선언
void createAccount();
void deposit();
void withdraw();
void displayAccounts();

#endif
```

헤더 파일로 분리하기

- **BankAccount.c**

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdbool.h>
#include "BankAccount.h"

//전역 변수 초기화로 메모리 공간이 할당됨
BankAccount accounts[MAX_ACCOUNTS]; //계좌 배열 생성
int accountCount = 0; //현재 계좌 수

//계좌 생성
void createAccount() {
    char accountNumber[ANO_LEN];  //입력(계좌번호)

    if (accountCount >= MAX_ACCOUNTS) {
        printf("더 이상 계좌를 생성할 수 없습니다.\n");
        return;
    }

    printf("계좌 번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);
```

헤더 파일로 분리하기

- **BankAccount.c**

```
    //중복 검사
    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            printf("이미 등록된 계좌입니다. 다시 입력하세요.\n");
            return;
        }
    }

    //중복 없을때 계좌 생성
    strcpy(accounts[accountCount].ano, accountNumber); //계좌번호 복사
    printf("계좌주: ");
    scanf("%s", accounts[accountCount].owner);
    accounts[accountCount].balance = 0; //잔고

    printf("결과: 계좌가 생성되었습니다. (계좌 번호: %s)\n",
        accounts[accountCount].ano);

    accountCount++;  //다음 인덱스로 증가
}
```

헤더 파일로 분리하기

- **BankAccount.c**

```c
//계좌 목록
void displayAccounts() {
    printf("********** 계     좌     목     록 **********\n");

    if (accountCount == 0)
    {
        printf("등록된 계좌가 없습니다.\n");
        return;
    }

    for (int i = 0; i < accountCount; i++)
    {
        printf("계좌 번호: %s, 계좌주: %s, 잔액: %d\n",
            accounts[i].ano, accounts[i].owner, accounts[i].balance);
    }
}
```

헤더 파일로 분리하기

- **BankAccount.c**

```c
//예금
void deposit() {
    char accountNumber[ANO_LEN];  //외부 입력(계좌번호)
    int amount;    //입금액
    bool found = false; //상태(계좌 찾음/못찾음)

    printf("입금할 계좌번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);

    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            found = true; //계좌 찾음

            printf("입금액: ");
            scanf("%d", &amount);
            if (amount < 0) {
                printf("올바른 금액이 아닙니다.\n");
                return; //즉시 종료
            }
```

헤더 파일로 분리하기

- **BankAccount.c**

```
            else {
                accounts[i].balance += amount;
                printf("정상 처리 되었습니다. 현재 잔액: %d\n",
                    accounts[i].balance);
                return;  //for문 탈출
            }
        }
    }

    if (!found) {
        printf("계좌를 찾을 수 없습니다.\n");
    }
}

//출금
void withdraw() {
    char accountNumber[ANO_LEN];  //외부 입력(계좌번호)
    int amount;  //출금액
    bool found = false; //상태(계좌 찾음/못찾음)

    printf("출금할 계좌번호(예: xx-xx-xxxx): ");
    scanf("%s", accountNumber);
```

헤더 파일로 분리하기

- **BankAccount.c**

```
    for (int i = 0; i < accountCount; i++) {
        if (strcmp(accounts[i].ano, accountNumber) == 0) {
            found = true; //계좌 찾음

            printf("출금액: ");
            scanf("%d", &amount);
            if (amount > accounts[i].balance || amount < 0) {
                printf("잔액이 부족하거나 올바른 금액이 아닙니다. "
                    "현재 잔액: %d\n", accounts[i].balance);
                return;
            }
            else {
                accounts[i].balance -= amount;
                printf("정상 처리 되었습니다. 현재 잔액: %d\n",
                    accounts[i].balance);
                return;  //for문 탈출
            }
        }
    }
    if (!found) {
        printf("계좌를 찾을 수 없습니다.\n");
    }
}
```

헤더 파일로 분리하기

- **Main.c**

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdbool.h>
#include "BankAccount.h"

int main()
{
    bool run = true;  //실행,종료
    int choice; //메뉴

    while (run)
    {
        printf("=====================================================\n");
        printf("1.계좌생성 | 2.예금 | 3.출금 | 4.계좌목록 | 5.종료\n");
        printf("=====================================================\n");
        printf("선택> ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            createAccount();
            break;
```

헤더 파일로 분리하기

- **Main.c**

```
            case 2:
                deposit();
                break;
            case 3:
                withdraw();
                break;
            case 4:
                displayAccounts();
                break;
            case 5:
                printf("프로그램을 종료합니다.\n");
                run = false;
                break;  //정상 종료
            default:
                printf("잘못된 입력입니다. 다시 선택하세요.\n");
            }
        }

        system("pause");  //콘솔창 닫힘 방지(파일 배포시)

        return 0;
}
```