프로그래밍 기초 과제
10 주차 실습

학과	컴퓨터공학과
학번	2022111120
이름	김지민
담당교수님	한인 교수님
제출일자	2023.05.14

| 실습문제 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SIZE 10
int main(){
    int iRandomNum;
    srand(time(0));

    int iArr[SIZE]={0};
    int i,j, iInputNum, iInputNumIdx;

    int iCount =0;

    while (iCount<10){
        iRandomNum=(rand()%100)+1;
        for (i=0;i<iCount;++i){
            if(iArr[i]==iRandomNum){
                break;
            }
        }
        if (i==iCount){//겹치는 수가 없다는 뜻이다!
            iArr[iCount]=iRandomNum;
            iCount=iCount+1;
        }
    }

    for(j=0;j<SIZE; ++j){
        printf ("%d ",iArr[j]);
    }

    printf("\n 상품번호를 입력하세요 : ");
    scanf("%d",&iInputNum);

    iInputNumIdx=iInputNum-1;//index 처리
    printf("상품 %d 번은 선반 %d 에 있습니다.",iInputNum,iArr[iInputNumIdx]);


    return 0;
}
```

| 코드 설명 :

1) 변수 설명

iInputNum: 확인하고 싶은 상품의 번호를 입력 받습니다.

iInputNumIdx : 입력한 수의 인덱스를 찾기 위해 만든 변수입니다. 사용자가 입력하는 상품의 번호는 자연수 체계이므로, 배열로 검색할 해당 상품 번호는 -1 를 해준 인덱스 값에 있기 때문입니다.

iCount : 10 개의 상품에 중복되지 않은 랜덤 번호를 부여하기 위해 만든 변수입니다.
2) while 반복문을 통한 상품 번호 부여하기
랜덤으로 설정할 상품의 장소는 10 개가 중복되면 안됩니다. 따라서 랜덤 생성을 몇 번 해야할지 확실히 예측할 수 없습니다. 중복되게 될 경우, 중복이 없을 때까지 반복해야하기 때문에 반복횟수를 알 수 없어서 for 가 아닌 while 을 이용했습니다.
그리고 while 을 종료하기 위한 조건으로 iCount <10 을 이용했습니다.
iCount 는 상품의 위치를 중복없이 랜덤하게 만들어냈을 때 마다 하나씩 증가하게 됩니다.

3) for 문을 통한 상품 장소의 중복 점검하기
10 개의 상품 장소를 생성할 때, 앞에서 생긴 수와 중복되지 않아야 합니다. 따라서 새로 만들어지는 상품 장소의 번호는 이 전에 만들어진 수와 모두 비교 해야합니다. 그래서 이미 만들어진 상품 장소 번호의 개수는 iCount 와 동일하므로, 조건식에 iCount 를 넣었습니다.
중복되지 않는다면 i 가 iCount 와 동일할 때까지 돌아가기 때문에 if 문을 통해서 i 와 iCount 비교를 통해 장소의 위치를 부여해줍니다.

| 출력 결과

```
45 29 7 63 33 77 70 61 96 24
상품번호를 입력하세요 : 5
상품 5번은 선반 33에 있습니다.
```

| 실습문제 2

```c
//과제 2 번
#include <stdio.h>
#define SIZE 5
int main(){
    int iArrX[SIZE]={0 };
    int iArrY[SIZE]={0 };
    int i,j,tempX,tempY;

   for (i=0;i<SIZE;++i){
        printf("x 와 y 좌표를 순서대로 입력하세요:");
        scanf("%d %d",&iArrX[i], &iArrY[i]);
    }

    for (j=0;j<SIZE-1;++j){
        for (i=0; i<SIZE-j-1; ++i){
            if (iArrX[i]>iArrX[i+1]){
                tempX=iArrX[i];
                iArrX[i]=iArrX[i+1];
                iArrX[i+1]=tempX;

                tempY=iArrY[i];
                iArrY[i]=iArrY[i+1];
                iArrY[i+1]=tempY;
```

```
            }
            else if (iArrX[i]==iArrX[i+1]){
                if(iArrY[i]>iArrY[i+1]){
                    tempX=iArrX[i];
                    iArrX[i]=iArrX[i+1];
                    iArrX[i+1]=tempX;

                    tempY=iArrY[i];
                    iArrY[i]=iArrY[i+1];
                    iArrY[i+1]=tempY;
                }
            }
            else {
                continue;
            }
        }
    }


    for (int k=0;k<SIZE;++k){
        printf("(%d,%d)",iArrX[k],iArrY[k]);
    }

    return 0;
}
```

| 코드 설명 :

1) 변수 설명

iArrX[SIZE]={0}; 과 iArrY[SIZE]={0};

x 좌표와 y 좌표를 각각 일차원 배열로 설정하여 5 개의 순서쌍을 저장할 수 있도록 설정했습니다.

tempX, tempY 는 버블 정렬을 진행하며 두 개의 원소를 바꿀 때 쓰기 위해 만든 변수입니다.

2) for 문을 이용한 버블 정렬 구현

첫 번째) 처음 원소를 5 까지 비교

두 번째) 처음 원소를 4 까지 비교

세 번째) 처음 원소를 3 까지 비교

네 번째) 처음 원소를 2 까지 비교

하는 방식이므로, 중첩 for 문을 이용해줍니다.

가장 바깥 for 문은 큰 단계로 몇 시행되는지를 의미하는 것으로, SIZE-1 만큼 반복합니다.

안의 for 문은 바깥 for 문의 시행 순서에 따라 원소를 1 부터 비교를 합니다. 따라서 SIZE-j-1 이 조건식이 됩니다.

그리고 if 문을 통해 x 좌표부터 먼저 비교해서, 크다면 x 와 y 좌표 모두 바꿔줍니다. 그리고 같다면 y 좌표를 비교하게 하여 자리를 바꿔 나갑니다.

| 출력 결과

```
int iArrX[SIZE]={4,3,2,5,2};
int iArrY[SIZE]={1,2,4,2,6}; (2,4)(2,6)(3,2)(4,1)(5,2)
```

이와 같이 정렬이 되는 모습을 확인할 수 있습니다.

| 추가 문제

```c
#include <stdio.h>
#define COLS 2
#define ROWS 3
int main(){
    int iAccountInfo[ROWS][COLS]={0};
    int iSelectMenu, inputAccount, inputMoney;
    int i,j;
    int checkAccount;
    checkAccount=0;

    while (1){
        printf("---menu---\n");
        printf("1.계좌개설 \n");
        printf("2.입금 \n");
        printf("3.출금 \n");
        printf("4.계좌 정보 전체 출력 \n");
        printf("5.프로그램 종료 \n");
        printf("선택:");
        scanf("%d",&iSelectMenu);

        if (iSelectMenu==1){
            printf("[계좌개설]\n");
            printf("계좌 ID:");
            scanf("%d",&inputAccount);
            printf("입금액:");
            scanf("%d",&inputMoney);
            if (inputMoney<0){
                printf("**계좌 개설 실패**\n");
                printf("입금액이 음수일 수 없습니다.\n");
                continue;
            }
            for (i=0;i<ROWS;++i){
                if (iAccountInfo[i][0]==0 ){
                    iAccountInfo[i][0]=inputAccount;
                    iAccountInfo[i][1]=inputMoney;
                    printf("**계좌 개설 성공!**\n");
```

```
                    printf("계좌 ID: %d ,잔액: %d
\n",iAccountInfo[i][0],iAccountInfo[i][1]);
                    checkAccount+=1;
                    break;
                }
                if (iAccountInfo[i][0]==inputAccount){
                    printf("**계좌 개설 실패**\n");
                    printf("이미 개설된 계좌입니다.\n");
                    break;
                }
                if (checkAccount==ROWS){
                    printf("**계좌 개설 실패**\n");
                    printf("%d 개 계좌가 이미 존재합니다.\n",ROWS);
                    break;
                }
            }

        }else if (iSelectMenu==2){
            printf("[입 금]\n");
            printf("계좌 ID:");
            scanf("%d",&inputAccount);
            printf("입금액:");
            scanf("%d",&inputMoney);

            if (inputMoney<0){
                printf("**계좌 개설 실패**\n");
                printf("입금액이 음수일 수 없습니다.\n");
                continue;
            }
            for(i=0;i<ROWS;++i){
                if(iAccountInfo[i][0]==inputAccount){
                    iAccountInfo[i][1]=iAccountInfo[i][1]+inputMoney;
                    printf("**입금 성공!**\n");
                    printf("계좌 ID: %d ,잔액: %d
\n",iAccountInfo[i][0],iAccountInfo[i][1]);
                    break;
                }else{
                    printf("**입금 실패**\n");
                    printf("계좌 id 가 존재하지 않습니다.\n");
                    break;
                }
            }
        }else if (iSelectMenu==3){
            printf("[출 금]\n");
            printf("계좌 ID:");
            scanf("%d",&inputAccount);
```

```
            printf("출금액:");
            scanf("%d",&inputMoney);

            for(i=0;i<ROWS;++i){
                if(iAccountInfo[i][0]==inputAccount){
                    iAccountInfo[i][1]=iAccountInfo[i][1]-inputMoney;
                    if (iAccountInfo[i][1]<0){
                        printf("**출금 실패!**\n");
                        printf("잔액이 부족합니다.\n");
                        iAccountInfo[i][1]=iAccountInfo[i][1]+inputMoney;
                        break;
                    }else {
                        printf("**출금 성공!**\n");
                        printf("계좌 ID: %d ,잔액: %d
\n",iAccountInfo[i][0],iAccountInfo[i][1]);
                        break;
                    }
                }else{
                    printf("**출금 실패**\n");
                    printf("계좌 id 가 존재하지 않습니다.\n");
                    break;
                }
            }
        }else if (iSelectMenu==4){
            for (i=0;i<ROWS;++i){
                if(iAccountInfo[i][0]==0){
                    break;
                }
                printf("계좌 : %d,
잔액 : %d\n",iAccountInfo[i][0],iAccountInfo[i][1]);
            }
        }else if (iSelectMenu==5){
            printf("프로그램을 종료합니다.\n");
            break;
        }else{
            printf("잘못된 입력입니다\n");
        }
    }

    return 0;
}
```

| 코드설명
이차원 배열로 계좌아이디 만큼의 행과, 2 개의 열을 설정해줍니다.
계좌 생성하기)

iAccountInfo[i][0]은 i 를 행만큼 반복했을 때, 계좌 아이디만 보게 해줍니다. 따라서 이 부분을 if 문을 통해서 존재하는지, 중복되는지, 10 개가 이미 찼는지 점검해줍니다.
입금액이 음수가 되면 에러 문자를 표시합니다.

입금하기)
마찬가지로 존재하는 아이디인지 확인하고, 음수를 입금하면 에러 메시지를 띄웁니다.

출금하기)
존재하는 아이디인지 확인하고, 잔액보다 출금액이 작은지 확인합니다. 만약 잔액보다 크다면 오류를 띄우고, 다시 원래 잔액으로 돌아갈 수 있도록 식을 작성해주었습니다.

계좌 출력하기)
계좌 아이디가 0 이 아니면 출력하고자 break 를 이용했습니다.

| 출력 결과

1 번 기능

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:1
[계좌개설]
계좌ID:120
입금액:150
**계좌 개설 성공!**
계좌ID: 120 ,잔액: 150
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:1
[계좌개설]
계좌ID:115
입금액:20
**계좌 개설 성공!**
계좌ID: 115 ,잔액: 20
```

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:1
[계좌개설]
계좌ID:115
입금액:30
**계좌 개설 실패**
이미 개설된 계좌입니다.
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:1
[계좌개설]
계좌ID:150
입금액:20
**계좌 개설 성공!**
계좌ID: 150 ,잔액: 20
```

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:1
[계좌개설]
계좌ID:150
입금액:20
**계좌 개설 성공!**
계좌ID: 150 ,잔액: 20
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:1
[계좌개설]
계좌ID:500
입금액:30
**계좌 개설 실패**
3개 계좌가 이미 존재합니다.
```

2 번기능

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:2
[입 금]
계좌ID:120
입금액:40
**입금 성공!**
계좌ID: 120 ,잔액: 190
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:2
[입 금]
계좌ID:300
입금액:40
**입금 실패**
계좌 id가 존재하지 않습니다.
```

3 번기능

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:3
[출 금]
계좌ID:120
출금액:90
**입금 성공!**
계좌ID: 120 ,잔액: 100
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:3
[출 금]
계좌ID:120
출금액:300
**출금 실패!**
잔액이 부족합니다.
```

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:3
[출 금]
계좌ID:120
출금액:20
**입금 성공!**
계좌ID: 120 ,잔액: 80
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:3
[출 금]
계좌ID:140
출금액:20
**출금 실패**
계좌 id가 존재하지 않습니다.
```

4 번기능

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:4
계좌 : 120, 잔액 : 80
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:4
계좌 : 120, 잔액 : 80
계좌 : 140, 잔액 : 30
```

```
---menu---
1.계좌개설
2.입금
3.출금
4.계좌 정보 전체 출력
5.프로그램 종료
선택:5
프로그램을 종료합니다.
```