**프로그래밍 기초 과제**

**11주차 실습**

|  |  |
| --- | --- |
| 학과 | 컴퓨터공학과 |
| 학번 | 2022111120 |
| 이름 | 김지민 |
| 담당교수님 | 한인 교수님 |
| 제출일자 | 2023.05.21 |

**| 해당 주차 내용** (5/10/수 수업 내용)

- 함수

정의 ) 하나의 독립적인 기능을 수행하는 별도의 모듈 (블록)이다.

언제 함수를 생성하는가? )

1) 독립적인 기능을 수행할 때, 별도의 함수로 분리한다.

2) 동일한 코드가 여러 곳에서 사용될 때, 동일한 코드를 하나의 함수로 분리한다.

\*메인 함수가 너무 길면 이해하기 어려우므로 독립 함수를 쓰는 것이다.

- 함수의 필요성과 장점

1) 코드의 중복을 막을 수 있다.

2) 한 번 작성된 코드는 여러 번 재사용할 수 있다.

3) 함수를 사용하면 전체 프로그램을 모듈로 분리할 수 있다.

4) 개발 과정이 쉬워지고 보다 체계적이게 되면서 유지보수도 쉬워진다.

🡪 모듈만 수정하면 되기 때문에!

- 함수의 정의 )

void (반환형) 함수이름 (매개변수) { 함수의 몸체 }

+ void는 리턴값이 없다.

- 반환형 )

함수가 처리를 종료한 후에 호출한 곳으로 반환하는 데이터의 유형을 뜻한다.

함수 이름

: 식별자에 대한 규칙만 따르면 된다.

: 함수의 기능을 암시하는 동사 or 동사 + 명사를 사용하면 좋다.

+ 소문자로 시작한다. (대문자로 시작하면 class 다)

- 함수의 호출

: 함수는 호출 관계가 있다. Calling 함수와 called 함수가 있다.

함수 안의 문장은 호출되기 전까지 전혀 실행되지 않는다. 메인 함수는 운영체제가 호출한다.

선언한 함수를 호출하게 되면, 현재 실행하고 있는 코드는 잠시 중단되고, 호출된 함수로 이동해서 함수 몸체 안의 문장이 순차적으로 실행된다.

호출된 함수의 실행이 끝나면 호출 위치로 돌아가서 잠시 중단 됐던 코드 실행이 재개된다.

- 인수와 매개변수

인수 : 호출 프로그램에 의하여 함수에 실제로 전달되는 값이다.

매개변수 : 인수를 전달받는 변수이다.

- 함수의 실행

called 함수를 아래에 쓰면 에러가 난다. (호출하는 함수보다 위에 있거나, 원형 <프로토타입>을 메인 함수 위에서 알려줘야 한다.

🡪 함수의 원형(프로토타입) : 함수를 선언했다고 컴파일러에게 함수에 대해 미리 알리는 것이다.

함수 헤더를 그대로 갖고 올라가서 세미콜론을 찍으면 된다.

함수는 정의 되기 전에 프로토타입을 쓰는 걸 원칙으로 한다.

- 함수의 원형 형식 : 함수의 반환 타입, 함수의 이름, 매개변수의 타입 리스트

**| 실습문제 1**

|  |
| --- |
| #include <stdio.h>  int is\_multiple(int n, int m);  int main(){  int n,m;  printf("첫 번째 정수를 입력하시오: ");  scanf("%d",&n);  printf("두 번째 정수를 입력하시오: ");  scanf("%d",&m);  if (n==0 || m==0){  printf("0은 입력할 수 없습니다.\n");  }  else {  if (is\_multiple(n,m)==1){  printf("%d은 %d의 배수입니다.\n",n,m);  }else if (is\_multiple(n,m)==0){  printf("%d은 %d의 배수가 아닙니다.\n",n,m);  }else {  printf("%d은 %d의 배수 입니다.\n",n,m);  }  }  return 0;  }  int is\_multiple(int n, int m){  int result;  int check;  check = n%m;  if (check==0){  result = 1;  }else{  result = 0;  }  return result;  } |

**| 코드 설명 :**

1) 첫 번째 if문

입력값이 0이면 안되므로 에러처리를 해줍니다.

그리고 0이 아니라면 배수 확인을 하므로 else 안에else-if문을 사용했습니다.

조건을 체크할 때, 함수를 호출합니다.

2) is\_multiple(int n, int m) 함수

배수인지 확인하는 함수로, 정수타입 n,m의 매개변수가 있습니다. 함수 내부에서는 배수인지 아닌지 판정한 후의 결과를 저장하는 result와, 배수인지 확인하기 위한 나머지 연산의 값을 갖고 있는 check 변수가 있습니다. 그래서 배수라면 result에 1을 부여하고, 배수가 아니면 0을 부여해서 result의 값을 반환합니다.

**| 출력 결과**

|  |
| --- |
|  |

**| 실습문제2**

|  |
| --- |
| #include <stdio.h>  int check\_alpha(char c);  int main(){  char inputChar;  int result;  printf("문자를 입력하세요 : ");  scanf("%c",&inputChar);  result = check\_alpha(inputChar);  if (result==1){  printf("%c는 알파벳 문자입니다.\n",inputChar);  }else {  printf("%c는 알파벳 문자가 아닙니다.\n",inputChar);  }  return 0;  }  int check\_alpha(char c){  int result;  if (c>=97&&c<=122){  result = 1;  }  else{  result = 0;  }  return result;  } |

**| 코드 설명 :**

**1) check\_alpha함수 생성**

**이 함수는 char 형식의 값을 매개변수로 받아서, 알파벳이라면 1을, 아니라면 0을 반환하는 함수입니다.**

**2) 메인함수에서 check\_alpha 함수를 통해 반환 받은 값을 저장할 변수 result를 선언해줍니다. 그리고 그 값을 if문을 통해 점검하며 알파벳 문자인지 아닌지를 출력합니다.**

**| 출력 결과**

|  |
| --- |
|  |

**| 추가문제**

|  |
| --- |
| #include <stdio.h>  #define COLS 2  #define ROWS 3  int iAccountInfo[ROWS][COLS]={0};  int checkAccount;  checkAccount=0;  void accountOpening (int inputAccount, int inputMoney);  void deposit (int inputAccount, int inputMoney);  void withDraw (int inputAccount, int inputMoney);  void printAccount();  int main(){  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("계좌ID:");  scanf("%d",&inputAccount);  printf("입금액:");  scanf("%d",&inputMoney);  accountOpening(inputAccount, inputMoney);    }else if (iSelectMenu==2){  printf("[입 금]\n");  printf("계좌ID:");  scanf("%d",&inputAccount);  printf("입금액:");  scanf("%d",&inputMoney);    deposit (inputAccount, inputMoney);    }else if (iSelectMenu==3){  printf("[출 금]\n");  printf("계좌ID:");  scanf("%d",&inputAccount);  printf("출금액:");  scanf("%d",&inputMoney);  withDraw(inputAccount,inputMoney);  }else if (iSelectMenu==4){  printAccount();  }else if (iSelectMenu==5){  printf("프로그램을 종료합니다.\n");  break;  }else{  printf("잘못된 입력입니다\n");  }  }    return 0;  }  void accountOpening (int inputAccount, int inputMoney){  int i,j;      for (i=0;i<ROWS;++i){  if (iAccountInfo[i][0]==0 ){  if (inputMoney<0){  printf("\*\*계좌 개설 실패\*\*\n");  printf("입금액이 음수일 수 없습니다.\n");  break;  }else {  iAccountInfo[i][0]=inputAccount;  iAccountInfo[i][1]=inputMoney;  printf("\*\*계좌 개설 성공!\*\*\n");  printf("계좌ID: %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("%d개 계좌가 이미 존재합니다.\n",ROWS);  break;  }  }  }  void deposit (int inputAccount, int inputMoney){  int i,j;  if (inputMoney<0){  printf("\*\*입금 실패\*\*\n");  printf("입금액이 음수일 수 없습니다.\n");  //*continue;*  }else{  for(i=0;i<ROWS;++i){  if(iAccountInfo[i][0]==inputAccount){  iAccountInfo[i][1]=iAccountInfo[i][1]+inputMoney;  printf("\*\*입금 성공!\*\*\n");  printf("계좌ID: %d ,잔액: %d \n",iAccountInfo[i][0],iAccountInfo[i][1]);  break;  }else{  printf("\*\*입금 실패\*\*\n");  printf("계좌 id가 존재하지 않습니다.\n");  break;  }  }  }  }  void withDraw (int inputAccount, int inputMoney){  int i,j;  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("계좌ID: %d ,잔액: %d \n",iAccountInfo[i][0],iAccountInfo[i][1]);  break;  }  }    if(inputAccount!=iAccountInfo[i][0]){  printf("\*\*출금 실패\*\*\n");  printf("계좌 id가 존재하지 않습니다.\n");  break;  }  }    }  void printAccount(){  int i,j;  for (i=0;i<ROWS;++i){  if(iAccountInfo[i][0]==0){  break;  }  printf("계좌 : %d, 잔액 : %d\n",iAccountInfo[i][0],iAccountInfo[i][1]);  }  } |

**| 코드 설명**

1) 함수 선언

accountOpening, deposit, withDraw, printAccount 총 4개의 void 타입의 함수를 생성했습니다. 함수를 호출하면, 호출된 함수에서 처리를 한 후 바로 print가 될 수 있도록 하고자 반환값이 없는 void타입의 함수를 선언했습니다.

printAccount를 제외한 모든 함수는 모두 inputAccount와 inputMoney를 매개변수로 하고 있습니다. 계좌를 생성할 때도, 입금할 때도, 출금할 때도 사용자가 계좌 ID를 입력해야하고, 입금 또는 출금액을 입력해야합니다. 따라서 두 값을 보내면 함수에서 전달받은 값을 이용하여 각각의 역할을 수행합니다.

반면 printAccount 함수는 전달하는 값도 없고, 매개변수도 없습니다. 그저 함수가 호출되어 실행되었을 때, 계좌 정보 배열을 확인하며 전체 계좌 정보 출력만 수행합니다.

2) 계좌 정보를 담는 배열을 전역 변수로 설정

4개의 생성된 함수는 모두 사용자의 계좌 정보가 담겨있는 배열에 접근해야 합니다. 만약 메인함수에 배열이 선언되어 있다면 새로 생성한 함수를 통해 조작한 내용은 전달할 수 없기 때문에 전역변수로 선언해주었습니다.

3) checkAccount 전역변수로 설정

계좌 생성할 때마다 checkAccount를 올려줘야합니다. 그래야 가득 찬 계좌 개수를 확인할 수 있는데, 함수 안에서 선언하게 될 경우, 호출이 끝나면 소멸하기 때문에 전역변수로 설정하여 누적하여 계산될 수 있도록 했습니다.

**| 출력 결과**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 계좌 개설 | | | | |
| 1. 성공 | 2. 실패 | | | |
|  | 이미 존재하는 ID | | 입금액이 음수 | 생성 가능 계좌 초과 |
|  | |  |  |
| 입금 | | | | |
| 성공 | 실패 | | | |
|  | 계좌아이디 없을 때 | | 입금액이 음수일 때 | |
|  | |  | |
| 출금 | | | | |
| 성공 | 실패 | | | |
|  |  | |  | |
| 출력 및 종료 | | | | |
|  | |  | | |