과제 보고서

과제 2 - 로또 당첨 프로그램

과목명: C프로그래밍
학 과: 소프트웨어학부
학 번: 2023203094
성 명: 김민준
제출일자: 2023-05-31

과제 보고서 - 로또 당첨 프로그램

I . 본문

1. 자신이 얻었다고 생각하는 점수 체크리스트

1. 사용자에게서 구매할 게임 수를 입력 받는다. 6개의 숫자를 입력받아 배열에 저장함.	V
2. 위의 항목을 구매한 만큼 반복	V
3. 사용자에게서 숫자를 입력 받을 때 중복된 값을 받지 않도록 예외 처리	V
4. 무작위로 당첨번호를 중복되지 않게 추첨하고, 화면에 출력(6개 + 보너스 숫자)	V
5. 추첨된 번호와 게임 번호를 대조하여, 당/낙첨 여부를 발표	V
6. 숫자는 오름차순으로 정렬(보너스 숫자는 맨 마지막에 출력)	V
7. 정렬에 대한 함수를 정의하여 구현	V
8. 구조체를 정의하여 구현	V
9. 2개 이상의 소스파일을 이용하여 구현(즉, extern 선언을 이용)	V
10. 보너스 점수	V

2. 실행 결과 스크린샷

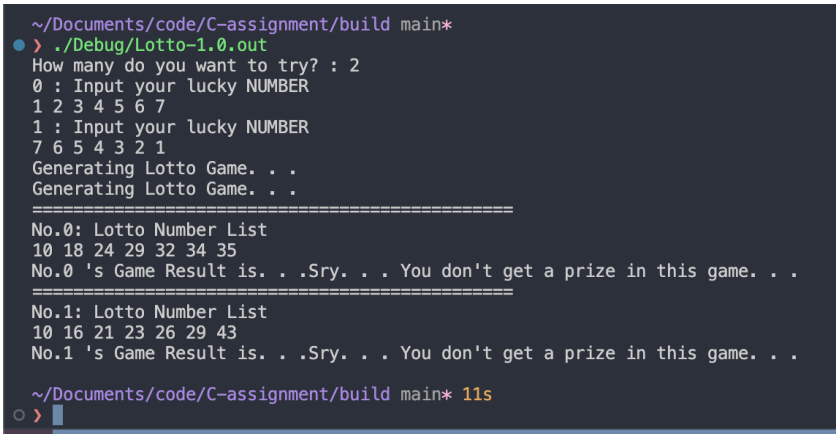


그림 1 전체 실행 결과

3. 구현 설명

Game.h

```
#pragma once

typedef struct Lotto_struct Lotto_struct;

/*
Lotto_struct: struct about lotto number
-> number: lotto number, this can store 5 numbers
-> extra_number: meaning as literally.
*/

struct Lotto_struct{
    int number[6];
    int extra_num;
};

struct Lotto_struct *newGame();
int* compareNumber(int* numbers, int extra, Lotto_struct* lotto);
void freeAllGame(Lotto_struct** lotto_ptr, int index);
```

#pragma once코드로 헤더파일을 딱 한번만 컴파일을 하도록 설정한다. struct Lotto_struct를 선언하여, Lotto 번호들을 모은 구조체를 선언한다. 이 구조체는 6자리의 number와 추가 번호인 extra_num로 구성되어 있다.

이제 Lotto번호를 뽑아줄 newGame 함수, 입력받은 수와 로또 번호를 비교해줄 compareNumber 함수, 할당된 메모리를 해제해줄 freeAllGame 함수를 헤더파일에 선언했다.

Game.c

1. **newGame Function**

```
#include"game.h"
#include"algorism.h"
#include<stdlib.h>

struct Lotto_struct* newGame(struct Lotto_struct* lotto) {
    srand((unsigned int)time(NULL)); //set Seed based on time

    lotto = (Lotto_struct*)calloc(1, sizeof(Lotto_struct)); //initialized new Lotto_struct

    for (int i = 0; i < 6; i++) {
        lotto->number[i] = rand() % 45 + 1; //set random number
    }
    lotto->extra_num = rand() % 45 + 1; //set extra number

    if (isSame(lotto->number)) {
        sleep(1);
        lotto = newGame(lotto); //re-generate Lotto Game
    }

    return lotto;
}
```

newGame 함수의 구조는 다음과 같다. 우선, srand함수를 통해time base의 seed를 설정하여 random값이 각자 다르게 나오도록 세팅한다. 함수의 인자값으로 들어온 Lotto_struct 타입의 lotto는 calloc을 통해 Lotto_struct size 1개를 할당 및 초기화를 한다.

lotto의 number에 rand()함수를 통해 난수를 생성하고 (45로 나눈 나머지+1) 연산을 통하여 1~45 사이의 수가 나오도록 초기화했다. extra_num도 마찬가지로 초기화해준다.

lotto의 number 안에서 겹치는 숫자가 없어야하기 때문에 isSame 함수(Algorism.h에 선언되어 있다.)를 호출하여 lotto의 숫자가 겹치는지 확인해준다. 겹친다면, 1초를 쉬고 다시 lotto struct를 만든다. 겹치지 않는다면 그대로 생성된 lotto를 반환해준다.

2. **compareNumber Function**

```
int compareNumber(int* numbers, int extra, Lotto_struct* lotto) {
    int count = 0;
    for (int i = 0; i < 6; i++) {
        if (numbers[i] == lotto->number[i]) { //check user input is same with lotto number
            count++;
        }
    }
    switch (count)
    {
    case 6: // 6 numbers correct
        return 1;
    case 5: // 5 numbers correct
        if (extra == lotto->extra_num) { //correct extra_number?
            return 2;
        }
        else {
            return 3;
        }
    case 4: // 4 numbers correct
        return 4;
    case 3: // 3 numbers correct
        return 5;
    default: // else
        return -1;
    }
}
```

Lotto_struct타입의 lotto, 입력받은 6개의 로또 숫자 배열인 number와 추가 숫자인 extra를 인자값으로 받는다. 입력받은 숫자와 랜덤값으로 생성된 lotto→number를 비교하여, 맞을때마다 count를 올린다. count에 따라 switch문으로 몇등인지 분기한다.

3. freeAllGame Function

```
void freeAllGame(Lotto_struct** lotto_ptr, int index) {
    for (int i = 0; i < index; i++) {
        free(lotto_ptr[i]);
    }
    free(lotto_ptr);
}
```

인자값으로 들어온 lotto 포인터를 전부 메모리 할당해제를 한다. 입력받은 게임 갯수만큼 lotto 포인터를 free함수를 통해 할당해제한다.

Algorism.h

```
#pragma once
#if _WIN32 || _WIN64
#define _CRT_SECURE_NO_WARNINGS
#include<windows.h>
#define sleep(time) Sleep((time*1000))
#else
#include<unistd.h>
#endif

int isSame(int* list); //check "is Same?"
int isExtraSame(int* list, int extra); //check extra number "is Same?"

void swap(int* a, int* b);
int* bubble_sort(int len, int* arr);
```

자주 쓰이는 알고리즘들을 모아 놓은 헤더파일이다. OS에 따라 불러오는 헤더파일을 다르게 구성하였다. Windows라면, Sleep함수를 Linux/Unix에서 쓰이는 sleep함수로 변환해주었다. Windows에선 Sleep의 인자값이 millisecond이므로, 1000을 곱해주었다.

list의 값이 같은지 확인하는 isSame함수, list의 값과 extra의 값이 같은지 확인하는 isExtraSame함수, 두 변수의 값을 바꿔줄 swap 함수, 배열을 오름차순으로 정렬하는 bubble_sort 함수를 선언했다.

Algorism.c

1. isSame Function

```
int isSame(int* list) {
    for (int i = 0; i < 6; i++) { //check some input number is same
        for (int j = i + 1; j < 6; j++) {
            if (list[i] == list[j]) {
                return 1;
            }
        }
    }
    return 0;
}
```

list내에서 같은 값이 있나 확인한다. 2중 for문을 통하여 모든 요소 하나씩 점검해보고 하나라도 같다면 1을, 모두 같지 않다면 0을 반환한다.

2. isExtraSame Function

```
int isExtraSame(int* list, int extra) {
    for (int i = 0; i < 6; i++) { //check some input number is same
        if (list[i] == extra) {
            return 1;
        }
    }
    return 0;
}
```

list내의 값과 extra 변수의 값이 같은지 확인한다. 같다면 1을, 같지 않다면 0을 반환한다.

3. swap Function

```
void swap(int* a, int* b) {
    int* temp = *b;
    *b = *a;
    *a = temp;
}
```

인자값으로 들어온 a와 b의 값을 바꿔주는 함수이다. temp 변수를 선언하여 b의 값을 받고 b에 a의 값을 넣어준다. 그 후, a에 temp값을 넣어주어 a와 b의 값을 바꾸어주었다.

4. bubbleSort Function

```
int* bubble_sort(int len, int* arr) {
    for (int i = 0; i < len; i++) {
        for (int j = 0; j < len; j++) {
            if (arr[j] > arr[j+1]) {
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }

    return arr;
}
```

bubble sort의 알고리즘을 구현한 함수이다. 배열의 1번째 수부터 접근을 하여 다음 번째의 수와 비교했을때, 크면 swap()함수를 통해 자리를 바꾸어 주고, 그렇지 않다면 swap을 하지 않고 계속 진행한다. 모든 정렬이 끝나면 정렬이 된 배열을 반환해준다.

main.c

```
#define _CRT_SECURE_NO_WARNINGS

//include Standard Header
#include<stdio.h>
#include<stdlib.h> // malloc from "stdlib.h" header
//include Custom Header
#include"game.h"
#include"algorism.h"

void input(int start, int end, int** input_list, int* extra); //return extra number;

int main(void) {
    Lotto_struct** lotto_ptr;
    int index = 0;
    int** input_num;
    int* input_extra;
    int result = -1;

    printf("How many do you want to try? : ");
    scanf("%d", &index);
    lotto_ptr = (Lotto_struct*)calloc(index, sizeof(struct Lotto_struct)); //set pointer of Lotto_struct
    input_num = calloc(index, sizeof(int)); //set length of 1st-dimension array
    for (int i = 0; i < index; i++) {
        input_num[i] = calloc(6, sizeof(int)); // set length of 2nd-dimension array
    }
    input_extra = calloc(index, sizeof(int));

    input(0, index, input_num, input_extra);

    for (int i = 0; i < index; i++) {
        printf("Generating Lotto Game. . .\n");
        lotto_ptr[i] = newGame(lotto_ptr[i]); //generate Lotto Game
        sleep(1);
    }

    for (int i = 0; i < index; i++) {
        result = compareNumber(input_num[i], input_extra, lotto_ptr); //check what prize player get

        int* sorted_number = bubble_sort(6, lotto_ptr[i]->number);

        printf("================================================\n");
        printf("No.%d: Lotto Number List\n", i); //print Lotto number List
        for (int j = 0; j < 6; j++) {
            printf("%d ", sorted_number[j]);
        }
        printf("%d\n", lotto_ptr[i]->extra_num);

        printf("No.%d 's Game Result is. . .", i);
        switch (result) //print result(like 1st, 2nd, 3rd. . .)
        {
        case 1: //1st
            printf("1st!!\n");
            break;
        case 2: //2nd
            printf("2nd!!\n");
            break;
        case 3: //3rd
            printf("3rd!!\n");
```

```
            break;
        case 4: //4th
            printf("4th :P\n");
            break;
        case 5: //5th
            printf("5th :<\n");
            break;
        default: //other
            printf("Sry. . . You don't get a prize in this game. . .\n");
            break;
        }
    }

    freeAllGame(lotto_ptr, index);
    for (int i = 0; i < index; i++) {
        free(input_num[i]);
    }
    free(input_extra);

    return 0;
}

void input(int start, int end, int** input_list, int* extra) {
    printf("%d : Input your lucky NUMBER\n", start);
    scanf("%d %d %d %d %d %d %d", &input_list[start][0], &input_list[start][1], &input_list[start][2], &input_list[start][3], &input_l

    //check all number in array "isSame" || check extra number is same with input nubmers array
    if (isSame(input_list[start]) || isExtraSame(input_list[start], extra)) {
        printf("Do not input the same number\n");
        input(start, end, input_list, extra); //run function again
    }

    start++;
    if (start != end) input(start, end, input_list, extra);
    else return;
}
```

위에 작성했던 헤더파일들을 main.c 맨 위에서 include해주었다.

1. Main Function
    - 몇번이나 Lotto를 추첨할지 입력받는다. 입력받은 값을 index에 저장한다.
    - lotto_struct를 index의 만큼 메모리를 할당하여 lotto_ptr를 초기화한다. 추후, lotto_sturct 포인터를 담는다.
    - input_num도 index*6만큼 2차원 배열로 메모리를 할당한다. input_extra도 int 크기만큼 메모리를 할당해준다.
    - 아래에 선언한 input()함수를 호출하여 값을 받아낸다. 이후 newGame()함수를 index만큼 호출해 무작위의 Lotto 번호를 제작하고 Lotto_ptr에 저장한다. 겹치는 숫자가 나오지 않도록 1초 기다려준다.
    - 입력받은 값과 Lotto번호를 compareNumber를 통해 비교하고 나온 등수에 따라 몇등인지 출력해준다. 동시에 Lotto 번호가 몇이 나왔는지 같이 출력한다. 이를 index만큼 반복해서 보여준다.
    - 할당했던 모든 포인터들을 해제한다.
2. input Function
    - index만큼 로또 번호를 반복 입력받는다.
    - 인자값으로 받은 input_list와 extra에 scanf로 얻은 로또 번호들을 넣어준다. 여기서 넣은 값들 중에 중복값이 있는지 isSame과 isExtraSame을 통해 검사한다. 중복값이 있다면 다시 input함수를 실행한다.
    - start가 end(=index)와 같지 않다면 input함수에 모든 인자값을 넣고 다시 호출한다. 여기서 start는 숫자를 입력받은 후 항상 1 증가한다.

CmakeList.txt

```
# 요구 CMake 최소 버전
CMAKE_MINIMUM_REQUIRED ( VERSION 3.0 )

# 프로젝트 이름 및 버전
PROJECT ( "Lotto" )
SET ( PROJECT_VERSION_MAJOR 1 )
SET ( PROJECT_VERSION_MINOR 0 )

# 빌드 형상(Configuration) 및 주절주절 Makefile 생성 여부
SET ( CMAKE_BUILD_TYPE Debug )
SET ( CMAKE_VERBOSE_MAKEFILE true )

# 빌드 대상 바이너리 파일명 및 소스 파일 목록
```

```
SET ( OUTPUT_ELF
        "${CMAKE_PROJECT_NAME}-${PROJECT_VERSION_MAJOR}.${PROJECT_VERSION_MINOR}.out"
        )
SET ( SRC_FILES
        src/algorism.c
        src/game.c
        src/main.c
        )

# 공통 컴파일러
set(CMAKE_C_COMPILER "/usr/local/Cellar/gcc/13.1.0/bin/gcc-13")
set(CMAKE_CXX_COMPILER "/usr/local/Cellar/gcc/13.1.0/bin/g++-13")

# 공통 헤더 파일 Include 디렉토리 (-I)
INCLUDE_DIRECTORIES ( include driver/include src/includes )

# 공통 컴파일 옵션, 링크 옵션
ADD_COMPILE_OPTIONS ( -g -Wall )
SET ( CMAKE_EXE_LINKER_FLAGS "" )

# 공통 링크 라이브러리 (-l)
LINK_LIBRARIES( )

# 공통 링크 라이브러리 디렉토리 (-L)
LINK_DIRECTORIES ()

# "Debug" 형상 한정 컴파일 옵션, 링크 옵션
SET ( CMAKE_C_FLAGS_DEBUG "-DDEBUG -DC_FLAGS" )
SET ( CMAKE_EXE_LINKER_FLAGS_DEBUG "-DDEBUG -DLINKER_FLAGS" )

# "Release" 형상 한정 컴파일 옵션, 링크 옵션
SET ( CMAKE_C_FLAGS_RELEASE "-DRELEASE -DC_FLAGS" )
SET ( CMAKE_EXE_LINKER_FLAGS_RELEASE "-DRELEASE -DLINKER_FLAGS" )

# 출력 디렉토리
SET ( CMAKE_RUNTIME_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYPE} )
SET ( CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYPE}/lib )
SET ( CMAKE_ARCHIVE_OUTPUT_DIRECTORY ${CMAKE_BUILD_TYPE}/lib )

# 빌드 대상 바이너리 추가
ADD_EXECUTABLE( ${OUTPUT_ELF} ${SRC_FILES} )
```

해당 프로젝트를 Visual Studio에서 생성하지 않아 직접 Cmake을 이용하여 빌드하였다. ~~(실은 Visual Studio에서 Cmake를 사용할 수 있긴하다.... 작업환경이 Mac이었기에 Cmake를 사용하였다.)~~

4. 빌드 및 실행

```
cmake -D CC=/usr/local/bin/gcc-13 -D CXX=/usr/local/bin/g++-13 . -B buildcmake -D CMAKE_C_COMPILER=/usr/local/bin/gcc-13 -D CMAKE_CXX_

cd build

make
```

다음과 같은 명령어를 입력하여 빌드하였다.

직접 플래그를 세팅하여 Gcc-13을 이용하도록 세팅하여 빌드하였다.

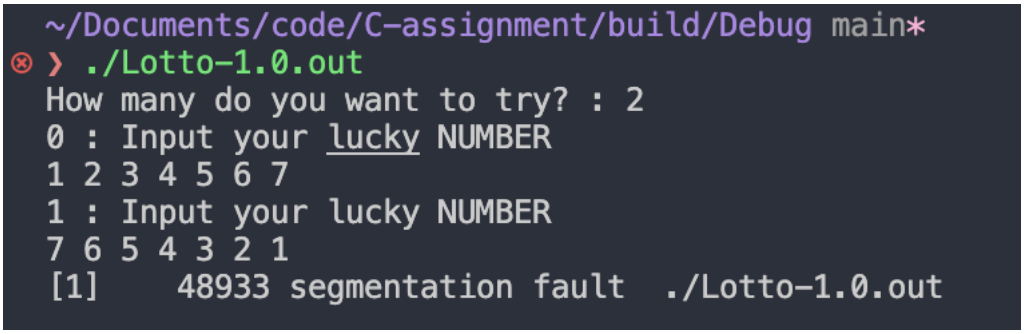
```
cd buld/Debug
sudo chmod +x Lotto-1.0.out
./Lotto-1.0.out
```

build/Debug에 생성된 out파일에 실행권한을 부여하고 실행하였다.

실행결과는 '2. 실행 결과 스크린샷' 과 같다.

II. 고찰

디버깅 도중 다음과 같은 에러가 났었다.



Segmentation fault 오류였다. 주로 stack overflow로 인하여 생기는 오류였다. 이 에러는 10번 중 3번 정도 발생하는 에러였다. 원활한 진행을 위해 오류 추적을 위해 어디서 Segmentation fault가 나는지 체킹해보았다.

발생되는 곳은 newGame()함수에서 발생하였다. 이는 재귀함수로 인한 수많은 함수호출과 struct 생성이 원인이었다. 수많은 함수 호출을 발생시키는 곳은 로또 번호가 같을때에서 재귀함수를 호출하는 곳밖에 없었기에 중단점을 설정하고 실행해보았다.

확인 결과 rand()함수는 srand(time(NULL))을 통해 time기반의 난수를 생성한다해도, 난수를 반복 생성할때 그 간격이 millisecond 차이에 관계 없이 같은 난수를 뽑아주는 것이 문제였다. 이는 생성된 로또 번호에서 똑같은 값이 나올때, 재귀함수를 호출하게 되고, 똑같은 값을 확인하고 재귀 호출하는 이 사이 간격이 고작 몇 millisecond밖에 나오지 않는다는 것이다. 재귀 호출 간격에서 millisecond가 아닌 일반 second 차이가 나기 위해 sleep()함수를 이용하여 Segmentation fault 오류를 해결하였다.

다음과 같은 에러를 해결하면서, 반복되는 재귀함수 호출과 struct 변수 선언은 힙과 스택에 모두 무리를 줄 수 있으며 오버플로우를 일으킬 수 있는 원인이 됨을 알게 되었다.