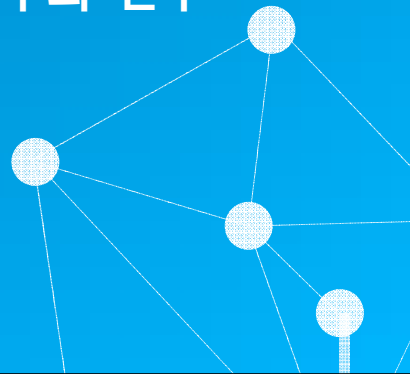


03

CHAPTER

함수와 변수



Target: 러시아 룰렛과 구구단 게임



• 러시아 룰렛(Russian Roulette)

- 게임 인원은 2명 이상
- 6연발 권총, 총알 수 지정(1~5)
- 시작하는 사람을 무작위로 선정
- 다음부터는 순서대로 진행.
- 방아쇠를 당겨 총에 맞으면 게임 종료



• 스피드 구구단

- 2부터 9사이의 곱셈 출제
- 총 10문제
- 문제를 푸는데 걸린 시간을 계산.
- 한 문제라도 틀린 경우 게임에서 진 것으로 판단
- 이긴 경우 전체 소요 시간을 이용한 점수 계산 및 출력



3장 학습 목표



- 함수의 개념과 사용 방법을 이해한다.
- 문제를 해결을 위한 함수의 설계 능력을 기른다.
- 함수에서 여러 개의 값을 반환할 수 있는 방법을 이해한다.
- 게임에 자주 사용되는 라이브러리 함수들을 이해하고 활용한다.
- 지역변수와 전역변수를 이해하고, static 지역변수를 이해한다.
- 변수의 생존 기간과 가시범위의 개념을 이해한다.
- 주어진 문제를 여러 개의 소스 파일에 나누어 구현할 수 있는 능력을 기른다.
- 전역 변수나 전역 함수를 static으로 처리하는 이유를 이해한다.

3

3.1 함수란?



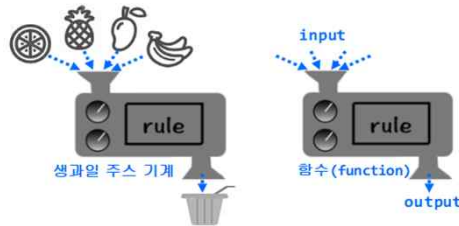
- 함수란?
- 라이브러리 함수
- 사용자 정의 함수
- 문제 해결과 함수

4

함수란?



- 생과일 주스 기계와 함수



함수의 장점

- 복잡한 문제를 쉬운 문제들로 나누어 해결하기 쉽도록 함
 - 구조적 프로그래밍, 모듈화, 캡슐화
- 같은 코드를 반복적으로 기술해야 하는 문제 해결
 - 코드 중복 최소화. 보완 등의 유지 보수가 쉬움
- 라이브러리 함수와 사용자 정의 함수

5

라이브러리 함수



- 라이브러리 함수의 예

함수 원형	헤더 파일	용도 및 사용 예
<code>int getche();</code>	<code><conio.h></code>	키보드에서 하나의 문자를 읽어서 반환함 예) <code>ch = getche();</code>
<code>double cos(double x)</code> ;	<code><math.h></code>	x에 대한 코사인 값을 계산하여 반환함 예) <code>val = cos(3.14);</code> <code>val = cos(x);</code>
<code>double pow(double x, int y)</code> ;	<code><math.h></code>	x의 y승을 계산하여 반환함 예) <code>result = pow(x, 10);</code> <code>result = pow(x, y);</code>
<code>int printf(const char* format [, argument]...)</code> ;	<code><stdio.h></code>	"format"에서 정하는 형태로 화면 출력 매개변수는 여러 개일 수 있음 예) <code>printf("game over !");</code> <code>printf("좌표(%d, %d)", x, y);</code>

6

• 라이브러리 함수 종류

- 표준 입출력
 - 화면 출력, 키보드 입력, 파일 입출력 함수 등
- 수학 연산
 - 다양한 수학 연산 함수들
- 문자열 처리
 - 문자열의 길이, 바꾸기, 연결하기 등 다양한 처리 함수 등
- 시간 처리
 - 현재 시각이나 처리 시간을 계산하는 함수 등
- 오류 처리
 - 오류 발생 검사 및 대처를 위한 함수들
- 데이터 검색 및 정렬
 - 배열 등 많은 데이터에서 원하는 것을 찾거나 정렬하는 함수들

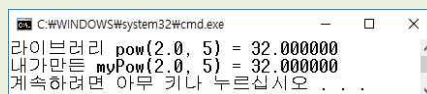
7

사용자 정의 함수

• 사용자 정의 함수 예

```
#include <math.h>           // pow()를 사용하기 위해 포함
double myPow(double x, int y){ // 사용자 정의 함수
{
    double result = 1.0;
    for (int i = 0; i < y; i++)
        result = result * x;
    return result;
}

void main()
{
    printf("라이브러리 pow(2.0, 5) = %1f\n", pow(2.0, 5));
    printf("내가 만든 myPow(2.0, 5) = %1f\n", myPow(2.0, 5));
}
```



8

문제 해결과 함수



- 함수는 문제 해결의 핵심
- 분할 정복(divide and conquer)
- 구구단 2~ 9단을 출력하는 다양한 방법들...
 - 방법 1: main()에서 모든 라인을 printf()로 직접 출력
 - 방법 2: main()에서 이중 반복문을 이용해 printf()로 출력
 - 방법 3: 구구단 2단에서 9단까지를 한꺼번에 출력하는 함수를 만들고 main()에서 이 함수를 한번 호출
 - 방법 4: 출력할 단을 입력으로 받아 그 단 만을 출력하는 함수를 만들고 main()에서 반복문으로 2~9단을 출력하도록 호출
 - 방법 5: 한 줄에 여러 단을 출력할 수 있는 함수를 구현

9

3.2 함수의 정의와 호출



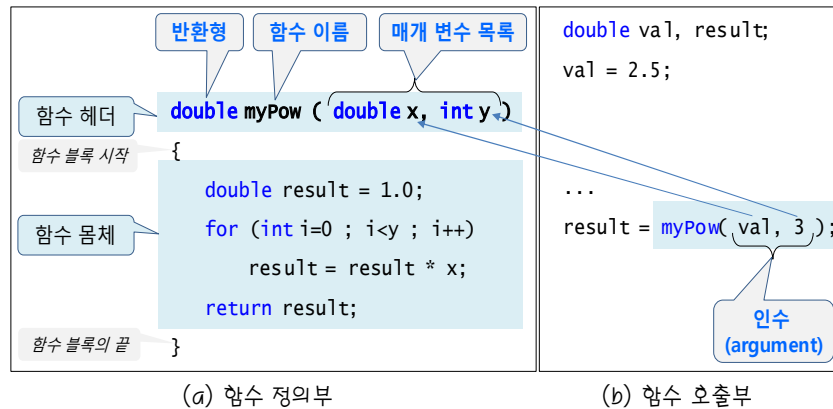
- 함수의 정의
- 함수의 호출

10

함수의 정의와 호출 구조



• 함수의 정의와 호출 구조



11

간단한 함수 구현의 예



```
// 두 int값을 더하고 결과를 반환하는 함수
int add ( int a, int b )
{
    int sum;                // 결과 값을 넣을 변수 선언
    sum = a + b;            // a 와 b를 더해 sum에 대입함
    return sum;             // 결과 값인 sum을 반환
}

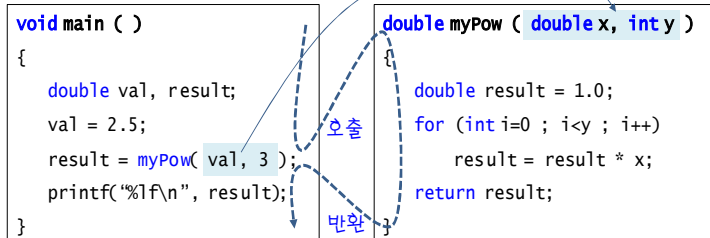
// 입력 값 n의 factorial을 구해 반환하는 함수
int factorial( int n )
{
    int result = 1;         // 결과값으로 사용할 변수 선언 및 초기화
    for(int i=1 ; i<=n ; i++) // i를 1~n까지 1씩 증가하면서 반복문 수행
        result *= i;       // result에 i를 곱함
    return result;          // 결과 값인 result를 반환
}
```

12

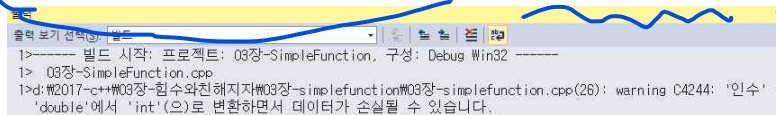
함수의 호출



- 인수(argument)가 매개 변수(parameter)로 복사됨



- 인수-매개 변수 자료형 불일치!!! : 예) factorial(4.0)



13

3.3. 함수 원형과 재사용



- 함수 원형(function prototype)
- 다른 소스 파일에서 함수 사용 방법
- Lab. 섭씨 화씨 계산 함수 구현
- Lab. 여러 개의 소스파일 포함

14

함수 원형(function prototype)



• 함수 원형(function prototype)

```
int factorial (int);           // factorial() 함수의 원형 선언
int add (int, int);           // add() 함수의 원형 선언

void main ( )
{
    int x=3, y=4, z;
    z = add (x, y);
    printf("x + y = %d\n", z );
    printf("x + 10 = %d\n", add(x,10));
    printf("5 + 10 = %d\n", add(5,10));

    z = factorial(x);
    printf("z! = %d\n", z);
    printf("5! = %d\n", factorial(5));
}
```

// 변수와 상수 전달
// 두 상수 전달
// 3!를 계산해 z에 대입
// z를 화면에 출력
// 5!를 화면으로 출력

```
x + y = 7
x + 10 = 13
5 + 10 = 15
3! = 6
5! = 120
```

15

다른 소스 파일에서 함수 사용 방법



방법 1 SimpleFnMain.cpp

```
#include <stdio.h>
int add(int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}

void main ()
{
    int x=3, y=4, z;
    z = add(x,y);
    printf("z=%d",z);
}
```

방법 2 SimpleFnMain.cpp

```
#include <stdio.h>
int add (int, int);
void main ()
{
    int x=3, y=4, z;
    z = add(x,y);
    printf("z=%d",z);
}

int add(int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}
```

방법 3 SimpleFn.cpp

```
int add(int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}
```

SimpleFnMain.cpp

```
#include <stdio.h>
int add(int, int);
void main ()
{
    int x=3, y=4, z;
    z = add(x,y);
    printf("z=%d",z);
}
```

방법 4 SimpleFn.h

```
int add (int, int);
```

SimpleFnMain.cpp

```
#include <stdio.h>
#include "SimpleFn.h"
void main ()
{
    int x=3, y=4, z;
    z = add(x,y);
    printf("z=%d",z);
}
```

16

Lab. 섭씨 화씨 계산 함수 구현



- tempConvert.cpp: 실제 함수 구현

```
double Celcius2Fahrenheit( double cels ) {
    double fahr = 32 + 180.0 / 100.0 * cels;
    return fahr;
}
double Fahrenheit2Celcius( double fahr ) {
    double cels = 100.0/180.0 * (fahr-32);
    return cels;
}
```

- tempConvert.h: 함수 원형 선언

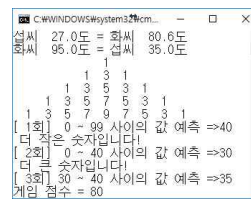
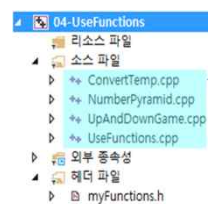
```
// tempConvert.h: 함수 원형 선언
double Celcius2Fahrenheit( double cels );
double Fahrenheit2Celcius( double fahr );
```

17

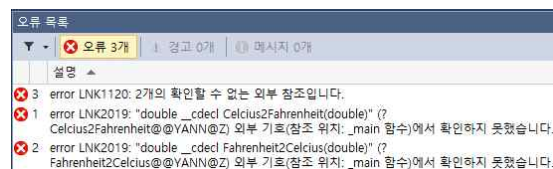
여러 개의 소스파일 포함



- 섭씨 화씨 변환, 숫자 파라미드, 숫자 맞추기 게임을 한 프로그램으로 만들기



- 만약 ConvertTemp.cpp가 프로젝트에 포함되지 않으면?
 - 링크 에러



18

3.4 함수 중복



- 함수 중복(function overloading)
- 자동 형 변환 문제

19

함수 중복(function overloading)



- **동일한 이름을 갖는 두 개 이상의 함수를 허용함**
 - 각각의 함수들은 매개변수의 자료형으로 구분
 - 자료형만 다른 동일한 연산을 같은 이름으로 처리할 수 있음
 - 매개변수가 같고 반환형만 다른 경우는 중복할 수 없음
 - 정적인 다형성(polymorphism)의 한 예

```
float add (float a, float b){  
    float sum;  
    sum = a + b;  
    return sum;  
}  
int add (int a, int b) {  
    int sum;  
    sum = a + b;  
    return sum;  
}  
  
double add(double a, double b){  
    double sum;  
    sum = a + b;  
    return sum;  
}  
void int main() {  
    int r1 = add(1, 2);  
    float r2 = add(1.0f, 2.0f);  
    double r3 = add(1.0, 2.0);  
}
```

20

자동 형 변환 문제



- 매개변수가 정확히 일치되지 않으면 자동 형 변환을 시도
- 이때 문제가 발생할 수 있다.

```
double dist( double x, double y ) { return sqrt(x*x + y*y); }  
int dist ( int x, int y ) { return x + y; }
```

```
int l1 = dist(3, 4);  
double l2 = dist(3.0, 4.0);
```

– 다음 호출은?

```
double l3 = dist(3, 4.0);
```

```
1 error C2666: 'dist' : 2개 오버로드에 비슷한 변환이 있습니다.  
2 IntelliSense: 오버로드된 함수 "dist"의 인스턴스 중 두 개 이상이 인수 목록과 일치합니다.  
함수 "dist(double x, double y)"  
함수 "dist(int x, int y)"  
인수 형식이 (int, double)입니다.
```

21

3.5 디폴트 매개변수와 인라인 함수



- 디폴트 매개변수(default parameter)
- 인라인 함수 (inline function)

22

디폴트 매개변수(default parameter)



- 디폴트 매개 변수를 잘 사용하여 선언한 함수 원형

```
void add(int p1, int p2, int p3=30);           // OK!  
void add(int p1, int p2=20, int p3=30);       // OK!  
void add(int p1=10, int p2=20, int p3=30);    // OK!
```

- 디폴트 매개 변수를 잘못 사용하여 선언한 함수 원형

```
void add(int p1, int p2=20, int p3);          // 컴파일 오류!  
void add(int p1=10, int p2, int p3=30);      // 컴파일 오류!
```

- 여러 함수를 구현한 효과가 있음

```
void f(int p1=1, double p2=2.0, char p3='a');  
  
void f(int p1, double p2, char p3);  
void f(int p1, double p2);  
void f(int p1);  
void f();
```

23

- 예: 선 그리기 (좌표 + 속성들)

```
void drawLine(int x1, int y1, int x2, int y2,  
              int width, int style, // 선의 두께와 선의 스타일  
              int r, int g, int b); // 선의 색상(r,g,b)
```

```
void drawLine(int x1, int y1, int x2, int y2,  
              int width=1, int style=PS_SOLID,  
              int r=255, int g=255, int b=255);
```

```
drawLine(10, 10, 20, 20); // 두께(1), 스타일(SOLID) 색상(255,255,255)  
drawLine(10, 10, 20, 20, 2); // 스타일(SOLID) 색상(255,255,255)  
drawLine(10, 10, 20, 20, 1, DASH); // 색상(255,255,255)  
drawLine(10, 10, 20, 20, 1, SOLID, 255,0,0); // 모두 지정
```

24

인라인 함수 (inline function)



- 매크로 함수의 문제: 기계적인 대치
 - 매개변수의 자료형에 관심 없음

```
#define SQUARE(x) (x*x)
```

일반화

```
...
```

```
SQUARE(y++); // -> y++*y++로 확장되어서 y의 값이 2번 증가
```

- 인라인 함수: 완전한 함수의 형태를 제공

```
inline int Square(int x) { return x*x; }  
inline double Abs(double x) { return (x>0) ? x : -x; }  
inline float Max(float a, float b) { return (a>b) ? a : b; }
```

- 헤더 파일에 두어야 함.

25

3.6 게임을 위한 라이브러리 함수



- 난수 발생
- 실행시간 측정
- 더 정밀한 실행시간 측정?

26

난수 발생



• 난수 발생

함수 원형	헤더 파일	용도 및 사용 예
<code>int rand(void);</code>	<code><stdlib.h></code> <code><cstdlib></code>	0 ~ RAND_MAX 사이의 임의의 정수 반환 예) <code>x = rand();</code>
<code>void srand(unsigned int seed);</code>	<code><stdlib.h></code> <code><cstdlib></code>	난수 생성기에 대한 시작 시드 값을 설정함. 예) <code>srand((unsigned)time(NULL));</code>
<code>time_t time(time_t* timer);</code>	<code><time.h></code> <code><ctime></code>	시스템 시간(1970년 1월 1일 자정 이후 경과된 시간(초))을 반환함. 예) <code>srand((unsigned)time(NULL));</code>

27

```
#include <stdlib.h>
#include <time.h>
inline int randCoin() { return rand() % 2; }
inline int randDice() { return rand() % 6 + 1; }
inline char randLChar() { return rand() % 24 + 'a'; }
inline char randUChar() { return rand() % 24 + 'A'; }
inline char randNum() { return rand() % 10 + '0'; }
void main()
{
    srand((unsigned)time(NULL));
    printf(" coin\tdice\tLChar\tUChar\tNum\n");
    for (int i = 0; i < 6; i++) {
        printf(" %s\t", randCoin()==0 ? "head" : "tail");
        printf("%2d\t", randDice());
        printf("%c\t", randLChar());
        printf("%c\t", randUChar());
        printf("%c\n", randNum());
    }
}
```

coin	dice	LChar	UChar	Num
tail	6	b	Q	1
head	6	v	C	0
head	4	m	B	7
head	6	e	G	4
tail	5	x	L	3
head	5	o	I	9

28

실행시간 측정

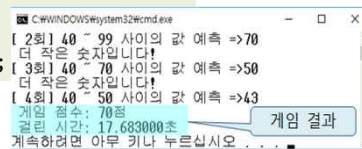


• 실행시간 측정

함수 원형	헤더 파일	용도 및 사용 예
<code>clock_t clock();</code>	<code><time.h></code> <code><ctime></code>	호출 시의 시스템 시각을 반환함 예) <code>clock_t start = clock();</code>

```
clock_t t1 = clock();
int score = playUpAndDown(43, 0, 99);
clock_t t2 = clock();
double duration = (double)(t2 - t1) / CLOCKS_PER_SEC;
```

```
printf(" 게임 점수: %d점\n", score);
printf(" 걸린 시간: %lf초\n", duration);
```



29

더 정밀한 실행시간 측정?



- 다음을 검색해 볼 것
- QueryPerformanceCounter 사용
- Chrono 사용
 - C++11에서 새로 추가된 시간 라이브러리
 - 기존의 C 런타임에서 제공하는 time 함수에 비해서 다양한 기능이 있고, 정밀도는 훨씬 높음.
 - chrono는 나노 밀리 초 단위도 측정할 수 있음.
 - VC11 이후에서 chrono는 STL에 포함

30

3.7 변수의 가시 범위와 생존 기간



- 변수의 세가지 속성
- 다양한 지역 변수와 전역 변수의 가시 범위
- extern 키워드
- static 키워드
- static 전역 변수, 함수

31

변수의 세가지 속성

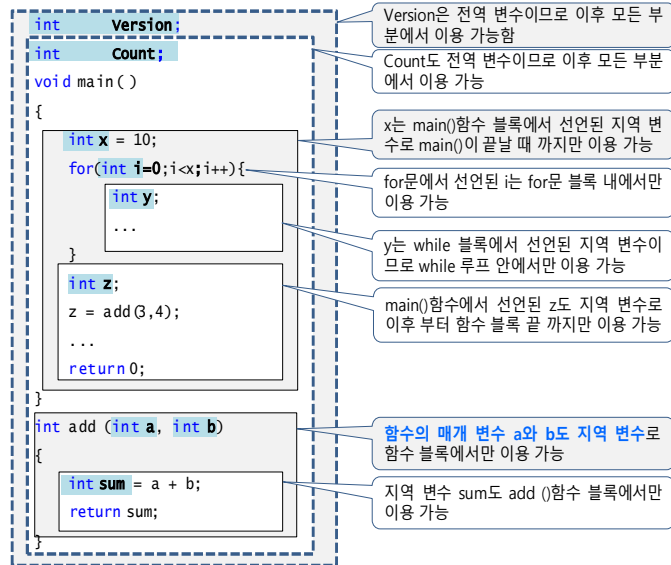


- 변수의 세가지 속성
 - 범위(visibility): 코드 내에서 변수가 의미 있는 영역
 - 생존 기간(lifetime): 만들어지는 시점과 소멸되는 시점
 - 연결(linkage): 외부에서 선언된 변수를 사용하는 방법

지역변수	전역변수
<ul style="list-style-type: none"> • 블록 범위(block scope) • 해당 블록 안에서만 의미가 있고 블록이 외부에서는 의미가 없음 • 선언 위치에서 만들어지고 해당 블록의 끝나면 소멸 • 변수가 선언된 블록과 그 블록 안에서 선언된 블록들에서만 보임 	<ul style="list-style-type: none"> • 파일 범위(file scope) • 선언 위치 이후부터 파일의 끝까지 어느 함수나 접근 가능 • 프로그램이 시작할 때 만들어지고, 프로그램이 끝나야 소멸 • 모듈간의 연관성을 최대한 줄이는 노력에 반하는 방법

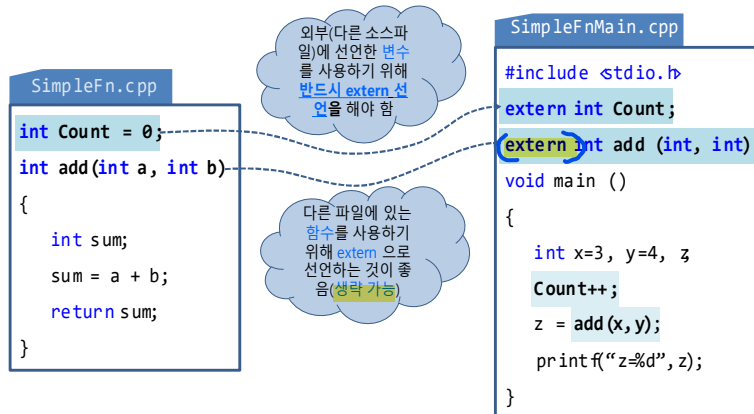
32

다양한 지역 변수와 전역 변수의 가시 범위



33

extern 키워드



34

static 키워드



- 세 가지 중요한 사용: **매우 중요**
 - 지역변수를 static으로 처리하는 경우
 - 전역변수나 함수를 static으로 처리하는 경우.
 - 클래스의 멤버변수나 함수를 static으로 처리하는 경우
- 정적 지역변수 **→ 커널!**
 - 프로그램의 실행 기간 동안 생존

```
int getNumberA()
{
    int count = 0;
    count++;
    return count;
}
```

```
int getNumberB()
{
    static int count = 0;
    count++;
    return count;
}
```

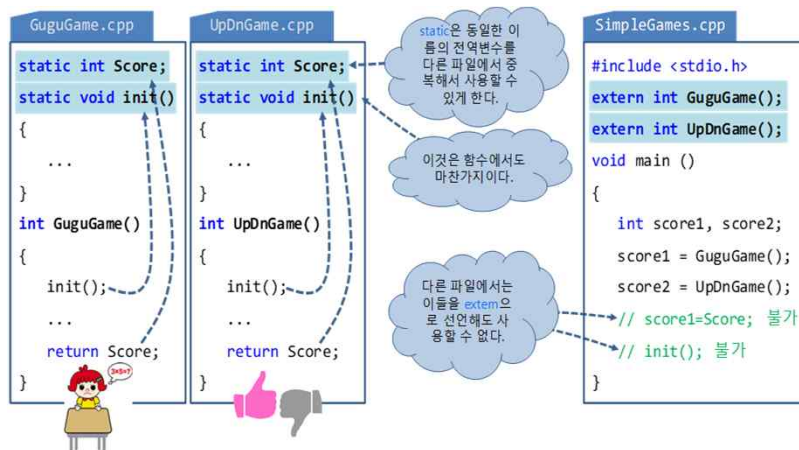
↓
계속 0이 됨

35

static 전역 변수, 함수



- 가시 범위의 문제 (Information hiding)



36

3.8 조금 살벌하고 긴장감 있는 게임



- 러시아 룰렛(디어 헌터) 게임
- 스피드 구구단 게임

37

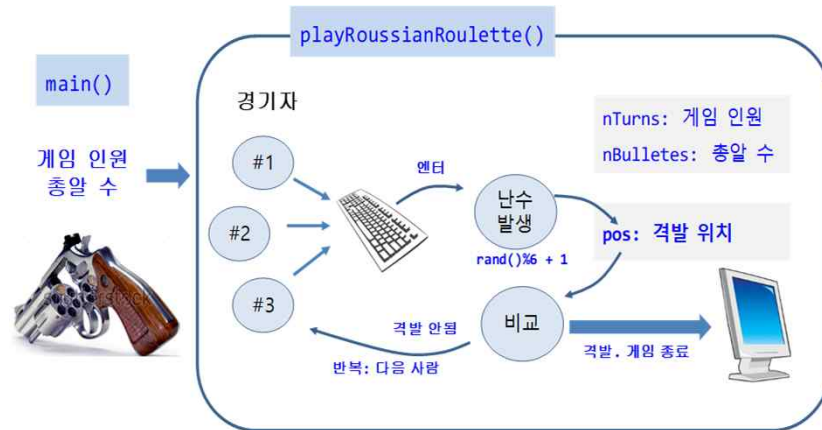
응용. 러시아 룰렛(디어 헌터)



- 러시아 룰렛(디어 헌터)
 - 환경
 - 게임 인원을 2명 이상.
 - 6연발 권총, 총알의 개수를 지정. 1발 이상 5발 이하.
 - 게임 방식
 - 영화에서는 총을 바닥에 놓고 돌려서 총구가 가리키는 사람부터 게임 시작
 - 프로그램에서는 랜덤 함수를 이용해 게임을 시작하는 사람을 정함.
 - 시작할 사람이 결정되면 다음부터는 순서대로 게임을 진행.
 - 종료 조건
 - 한 사람이 방아쇠를 당겨 총에 맞으면 게임 종료
 - 그렇지 않으면 탄창을 다시 무작위로 돌리고 다음 사람이 방아쇠를 당김.
 - 게임은 함수로 구현
 - 총알을 맞은 사람의 번호를 반환
 - 게임 함수는 파일을 분리하여 구현

38

• Russian Roulette Game



39

```
extern int playRoussianRoulette(int nTurns=2, int nBullets=1);
```

```
#include "RoussianRoulette.h"
void main()
{
    int nTurn, nBullets;
    srand((unsigned)time(NULL));
    printf("게임 인원 (예:2) ==> ");
    scanf("%d", &nTurn);
    printf("총알 개수 (6미만) ==> ");
    scanf("%d", &nBullets);
    getchar();

    int bang = playRoussianRoulette(nTurn, nBullets);
    printf("\n -----> %d번 참가자가 총에 맞았습니다.\n", bang);
}
```

```
C:\WINDOWS\system32\cmd.exe
게임 인원 (예:2) ==> 3
총알 개수 (6미만) ==> 2

총을 돌렸습니다. 3번부터 시작합니다.
[3번] 탄창을 무작위로 돌렸습니다.
엔터를 누르면 격발됩니다...
휴~~~~~ 살았습니다!!!

[1번] 탄창을 무작위로 돌렸습니다.
엔터를 누르면 격발됩니다...
휴~~~~~ 살았습니다!!!

[2번] 탄창을 무작위로 돌렸습니다.
엔터를 누르면 격발됩니다...
빵~~~~~ !!!

-----> 2번 참가자가 총에 맞았습니다.
계속하려면 아무 키나 누르십시오 . . .
```

40

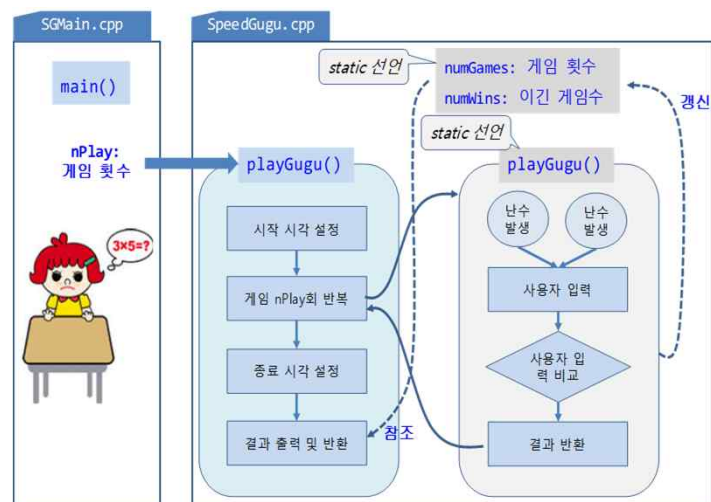
응용: 스피드 구구단



- 조건
 - 문제로 나오는 곱셈은 2부터 9사이의 수로 이루어짐.
 - 전체 문제는 10번 출제, 문제를 푸는데 걸린 시간을 계산.
 - 한 문제라도 틀린 경우 게임에서 진 것으로 판단.
 - 그렇지 않으면 전체 시간을 이용해 적절한 점수를 계산.
 - 소스 파일은 작은 단위로 분리하여 구현.

```
extern double tElapsed; // 게임 소요 시간
extern double playSpeedGugu(int nPlay);
```

41



42



```
#include "SpeedGugu.h"
static int NumGames = 0; // 전체 시도 횟수
static int NumWins = 0; // 맞힌 횟수
static double Score = 0; // 점수
double tElapsed = 0; // 게임 소요시간
static bool playGuguOnce() {...}
double playSpeedGugu(int nPlay) { ... }
```

C:\WINDOWS\...
문제 1): 7 x 2 = 14
문제 2): 5 x 3 = 15
문제 3): 9 x 2 = 18
문제 4): 8 x 2 = 16
문제 5): 4 x 7 = 28
문제 6): 7 x 9 = 63
문제 7): 9 x 6 = 54
문제 8): 5 x 9 = 45
문제 9): 5 x 3 = 15
문제 10): 5 x 9 = 45
점수 = 49.9점(총 25.1초)

```
#include "SpeedGugu.h"
void main()
{
    srand((unsigned)time(NULL));
    int nPlay = 10;
    printf("[스피드 구구단 게임]\n\n");
    ...
    double score = playSpeedGugu(nPlay);
    printf("\n점수 = %.1f점(총 %.1f초)\n", score, tElapsed);
}
```

C:\WINDOWS\system32\cmd.exe
[스피드 구구단 게임]
당신의 구구단 실력을 테스트하세요.!!!
10번 테스트 하셨습니다.
크게 심호흡을 하시고...
준비되면 엔터를 누르세요...

43

3장 요약문제, 연습문제, 실습문제



44



감사합니다!