C++_배열, 함수, 포인터

Visual Studio 2022

배열(Array)

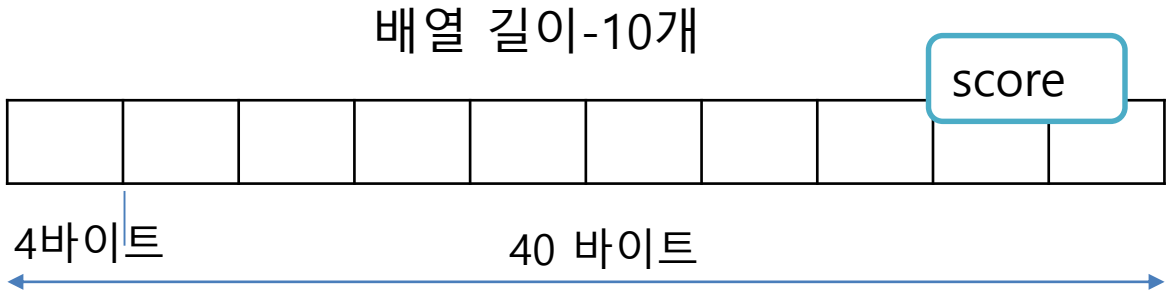
- 배열이란?

  여러 개의 연속적인 값을 저장하고자 할 때 사용하는 자료형이다.

  배열 변수는 [ ]안에 설정한 값만큼 메모리를 할당하여 저장한다.

- 배열 변수의 선언과 사용

  int score[10];



배열(Array)

- 문자형 배열 관리 비교

구분	C언어	C++
초기화(생성)	**char** arr[3][10] = {'a', 'b', 'c'}	**string** arr[ ] = {'a', 'b', 'c'}
배열의 크기	sizeof(arr) / sizeof(arr[0])	size(arr) 함수

- 범위기반 for문

```
for(자료형 변수이름 : 배열이름){
        출력 : 변수
}
```

```
for(string a : arr){
    printf("%s", a);
}
```

배열(Array)

- 문자형 배열 관리

```
//문자열 배열 관리
//c언어
char season[4][10] = {"봄", "여름", "가을", "겨울"};
//printf("%s ", list[0]);
int len = sizeof(season) / sizeof(season[0]);
for (int i = 0; i < len; i++) {
    printf("%s ", season[i]);
}

//c++
string carts[] = { "라면", "쌀", "생수", "화장지" };

//배열의 크기
cout << "\n배열의 크기(길이): " << size(carts) << endl;

//2번 요소 조회
cout << carts[2] << endl;
```

배열(Array)

- 문자형 배열 관리

```
//요소 수정
carts[1] = "빵";

//전체 출력
for (int i = 0; i < size(carts); i++) {
    cout << carts[i] << " ";
}
cout << endl;

//향상된 for문 - for(자료형 변수 : 배열이름){}
for (string cart : carts) {
    cout << cart << " ";
}
```

```
봄 여름 가을 겨울
배열의 크기(길이): 4
생수
라면 빵 생수 화장지
라면 빵 생수 화장지
```

배열(Array)

- 정수형 배열 관리

```
/*int arr[3] = {0};
arr[0] = 1;
arr[1] = 2; */

int arr[3] = { 1, 2, 3 };

arr[1] = 5;  //수정

for (int i = 0; i < size(arr); i++) {
    cout << arr[i] << " ";
}
cout << endl;

//범위 기반 for문
for (int a : arr)
    cout << a << endl;
```

배열(Array)

- 정수형 배열의 연산

```
int array[] = { 90, 80, 75, 100 };
int total = 0;
float average;

cout << array[0] + array[1] << endl; //170

//합계
for (int i = 0; i < size(array); i++) {
    total += array[i];
}
cout << "total = " << total << endl;

//평균 = 합계 / 개수
average = (float)total / size(array);
cout << fixed;  //소수 자리수 설정
cout.precision(1);
cout << "average = " << average << endl;
```

함수(function)

❖ 함수(Function)란?

- 하나의 기능을 수행하는 일련의 코드이다.(모듈화)
- 함수는 이름이 있고, 반환값과 매개변수가 있다.(함수의 형태)
- 하나의 큰 프로그램을 작은 부분들로 분리하여 코드의 중복을 최소화하고, 코드의 수정이나 유지보수를 쉽게 한다.(함수를 사용하는 이유)

  - 모든 코드를 main(){...} 함수 내에서 만들면 중복 및 수정의 복잡함이 있음

❖ 함수의 종류

- 내장 함수 – 수학, 시간, 문자열 함수 등
- 사용자 정의 함수 – 사용자(개발자)가 직접 만들어 사용하는 함수

```
반환자료형 함수이름(매개변수)
{
    구현 코드
}
```

```
int getArea(int x, int y)
{
    return x * y
}
```

함수(function)의 유형

- 반환자료형이 없는 함수 - void

```
#include <iostream>
using namespace std;

void printGuguDan(int dan) {
    for (int i = 1; i <= 9; i++) {
        cout << dan << " x " << i << " = " << dan * i << endl;
    }
}

int main() {

    cout << "구구단 " << endl;
    printGuguDan(3);

    return 0;
}
```

함수(function)의 유형

- 반환 자료형이 있는 함수 – **return** 키워드 사용

```
//제곱수 계산 함수
int square(int x)
{
    return x * x;
}

//절대값 계산 함수
int myAbs(int x)
{
    if (x < 0)
        return -x;
    else
        return x;
}
```

함수(function)의 유형

- 반환 자료형이 있는 함수 – **return** 키워드 사용

```
//두 수의 합 계산 함수
int add(int x, int y)
{
    return x + y;
}

int main()
{
    //square() 호출
    int value1 = square(4);
    cout << "제곱수: " << value1 << endl;

    //myAbs() 호출
    int value2 = myAbs(-5);
    cout << "절대값: " << value2 << endl;

    //add() 호출
    int value3 = add(10, 20);
    cout << "두 수의 합: " << value3 << endl;

    return 0;
}
```

변수의 적용 범위 – 전역 변수, 지역변수

- 전역 변수(static variable)

  전역 변수의 메모리 생성 시점 - main() 위 전역 공간에 생성
  전역 변수의 메모리 소멸 시점: - 프로그램이 종료되었을 때

- 지역 변수(local variable)

  지역 변수의 메모리 생성 시점 - 제어문이나 함수의 블럭안에서 생성
  지역 변수의 메모리 소멸 시점: - 호출된 후 블록을 벗어났을때

```
//int x = 10;  //전역 변수
//int y = 10;

void click() {
    int x = 10; //지역 변수
    static int y = 10; //지역변수가 전역변수화 함

    x++;
    y++;

    cout << "x = " << x << ", y = " << y << endl;
}
```

변수의 적용 범위 – 정적 변수

- 정적 변수(static variable)
  - 선언된 함수가 종료하더라도 그 값을 계속 유지하는 변수
  - **static** 키워드를 붙임

정적 변수의 메모리 생성 시점 - 중괄호 내에서 초기화될때
정적 변수의 메모리 소멸 시점: - 프로그램이 종료되었을 때

```
int main()
{
    //click() 여러 번 호출
    click();
    click();
    click();
    click();

    return 0;
}
```

```
x=11, y=11
x=11, y=12
x=11, y=13
x=11, y=14
```

기본 매개 변수

- 기본 매개 변수(default parameter)
  - 기본값이 제공된 함수의 매개 변수이다.
  - 호출시 생략하면 기본값이 전달됨.

```
void take(int busNumber, int fee = 1500) {
    cout << busNumber << "번 버스 : "
         << "요금 " << fee << "원\n";
}

int main()
{
    take(101); //생략하면 기본 요금이 전달됨
    take(500, 1500);
    take(1700, 2000);

    return 0;
}
```

```
101번 버스 : 요금 1500원
500번 버스 : 요금 1500원
1700번 버스 : 요금 2000원
```

참조에 의한 호출

- 참조자란?

  참조형을 레퍼런스라고도 하는데, 기존의 메모리공간에 별명(alias)을 붙이는 방법을 말한다. (포인터와 유사함)

  하나의 변수에 여러 개의 이름을 붙이는 것을 말한다.

  자료형& 참조변수명 (&는 참조 연산자)으로 사용한다.

- 참조자의 활용

  ① 함수의 매개변수로 사용하기 위해(★중요 ★)

  ② 함수의 반환형으로 사용하기 위해

참조에 의한 호출

- 참조자 실습

```
//참조 연산자 사용
int n = 1;
int& x = n;
int& y = n;

cout << "x = " << x << endl; //1
cout << "y = " << x << endl; //1

x = 3;
cout << "x = " << x << endl; //3
cout << "y = " << x << endl; //3
```

참조에 의한 호출(call-by-reference)

- 참조에 의한 호출

```
void swapVal(int a, int b);
void swapRef(int& a, int& b);
void swapRef2(int* a, int* b);
int main()
{
    //참조(&) - 미리 정의된 변수의 실제 이름 대신 사용하는 이름(별칭-alias)
    int x = 10, y = 20;

    cout << "값에 의한 호출\n";
    swapVal(x, y);
    cout << "x = " << x << ", y = " << y << endl;

    cout << "참조에 의한 호출\n";
    swapRef(x, y);
    cout << "x = " << x << ", y = " << y << endl;

    cout << "포인터에 의한 호출\n";
    swapRef2(&x, &y);
    cout << "x = " << x << ", y = " << y << endl;
    return 0;
}
```

참조에 의한 호출(call-by-reference)

- 참조자 실습

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

void swapRef(int& a, int& b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

```
void swapRef2(int* a, int* b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

```
값에 의한 호출
x = 10, y = 20
참조에 의한 호출
x = 20, y = 10
```

인라인 함수(inline Function)

- 인라인 함수란?

  inline함수란 **함수 호출 오버헤드**로 인한 프로그램의 실행 속도 저하를 막기 위한 기능으로 인라인 함수의 코드를 그대로 삽입하여 **함수 호출**이 일어나지 않게 한다.

  (오버헤드란 어떤 명령어를 처리하는데 소비되는 간접적, 추가적인 컴퓨터 자원을 의미한다.)

  - 사용 예시

  ```
  inline add(x, y) {return x + y}
  ```

인라인 함수(inline Function)

```
//인라인 함수 정의
inline int square(int x) { return x * x; }
inline int odd(int x) { return x % 2; }
int main()
{
    //제곱수 계산하기
    int val = square(6);
    cout << "제곱수: " << val << endl;

    //홀수의 합 계산
    int sum = 0;
    for (int i = 0; i <= 10; i++) {
        if (odd(i)) { //if(1){}, 1=true
            sum += i; //1+3+5+7+9
        }
    }
    cout << "합계: " << sum << endl;

    return 0;
}
```

표준 라이브러리 함수(function)

❖ 내장 함수 – 표준 라이브러리 함수

Standard library header <ctime>

This header was originally in the C standard library as `<time.h>`.

This header is part of the C-style date and time library.

Macro constants

CLOCKS_PER_SEC	number of processor clock ticks per second (macro constant)
NULL	implementation-defined null pointer constant (macro constant)

Types

clock_t	process running time (typedef)
size_t	unsigned integer type returned by the `sizeof` operator (typedef)
time_t	time since epoch type (typedef)
tm	calendar time type (class)

수학 함수(function)

✓ 수학 관련 함수 – <cmath>를 include 해야 함

```
#include <iostream>
#include <cmath>
using namespace std;

//수학 관련 내장 함수 사용하기
int main()
{
    //반올림
    cout << "2.54 반올림: " << round(2.54) << endl;
    cout << "2.45 반올림: " << round(2.45) << endl;

    //내림
    cout << "3.3 내림: " << floor(3.3) << endl;

    //절대값
    cout << "8 절대값: " << abs(8) << endl;
    cout << "-8 절대값: " << abs(-8) << endl;

    //거듭제곱
    cout << "2의 4제곱: " << pow(2, 4) << endl;

    //제곱근
    cout << "16의 제곱근: " << sqrt(16) << endl;

    return 0;
}
```

시간 함수(function)

✓ 시간 관련 함수 – <ctime>을 include 함

```
#include <iostream>
#include <ctime>
#include <thread> //this_thread::sleep_for()
using namespace std;

int main()
{
    //NULL 대신 nullptr 사용
    time_t now = time(nullptr); //현재 시간

    //1970.1.1 자정 이후부터 현재시간까지 초로 측정
    cout << now << "초\n";
    cout << now / (24 * 60 * 60) << "일\n";
    cout << now / (365 * 24 * 60 * 60) << "년\n";
```

^시간 함수(function)

✓ 수행 시간 측정하기

```
time_t start, end;

//start = time(nullptr);
time(&start); //시작 시간

for (int i = 1; i <= 10; i++) {
    cout << i << endl;
    //0.5초 간격으로 대기
    this_thread::sleep_for(chrono::milliseconds(500));
}

//end = time(nullptr);
time(&end);  //종료 시간
cout << "수행시간: " << (end - start) << "초\n";
```

```
1753313369초
20292일
55년
1
2
3
4
5
6
7
8
9
10
수행시간: 5초
```

^시간 함수(function)

✓ 수행 시간 측정하기(실수로 측정)

```
//수행 시간 측정(실수)
time_t start, end;
double elapsedTime;

start = clock();
for (int i = 1; i <= 10; i++) {
    cout << i << endl;
    //0.5초 간격으로 대기
    this_thread::sleep_for(chrono::milliseconds(500));
}

end = clock();
//CLOCKS_PER_SEC - 초 단위 변환 상수
elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;

cout << fixed;
cout.precision(2);
cout << "수행시간: " << elapsedTime << "초\n";
```

```
1
2
3
4
5
6
7
8
9
10
수행시간: 5.07초
```

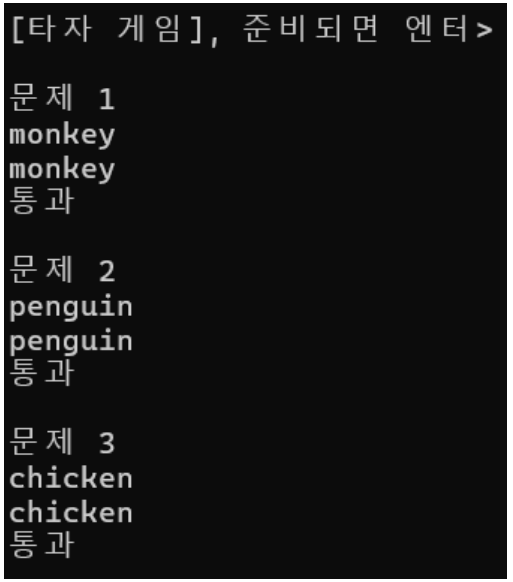
rand() 함수

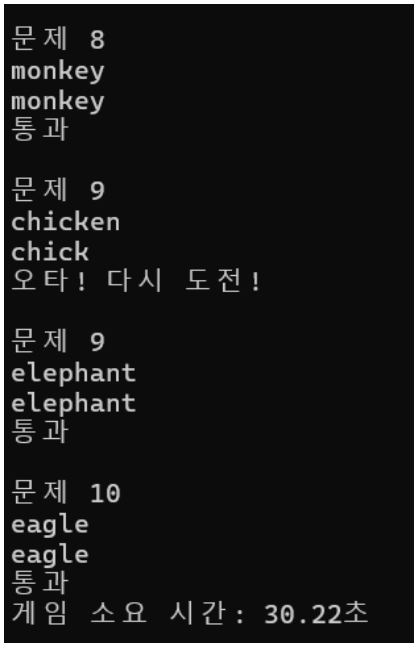
✓ 문자열 랜덤하게 추출하기

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
    srand(time(nullptr)); //시드 설정

    int randVal = rand();
    cout << randVal << endl; //랜덤수 출력

    //문자 추출
    string seasons[] = { "봄", "여름", "가을", "겨울" };
    //cout << seasons[3] << endl; //겨울
    cout << size(seasons) << endl;  //4

    int randIdx = rand() % size(seasons); //난수 생성
    cout << seasons[randIdx] << endl;
    return 0;
}
```

English Typing Game

- **게임 방법**
  - 게임 소요 시간을 측정함
  - 컴퓨터가 저장된 영어 단어를 랜덤하게 출력함
  - 사용자가 단어를 따라 입력함
  - 출제된 단어와 입력한 단어가 일치하면 "통과!", 아니면 "오타! 다시 도전"
    출력횟수는 총 10번이고, 끝나면 "게임을 종료합니다." 출력





Typing Game

- 영어 타자 게임

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <string>
using namespace std;

int main()
{
    string words[] = { "ant", "bear", "chicken", "elephant",
        "eagle", "horse", "lion", "monkey", "penguin", "tiger" };
    string question;  //문제
    string answer;    //사용자 입력 답
    int n = 1;        //문제 번호
    clock_t start, end; //시작, 종료시간
    double elapsedTime; //소요시간
```

Typing Game

- 영어 타자 게임

```
srand(time(nullptr)); //시드 설정
cout << "[타자 게임], 준비되면 엔터> ";
cin.ignore();  //버퍼 정리

start = clock();  //시작
while (n <= 10) {
    cout << "\n문제 " << n << endl;
    int rndIdx = rand() % size(words);
    question = words[rndIdx];
    cout << question << endl;
    getline(cin, answer);

    if (answer.compare(question) == 0) {
        cout << "통과\n";
        n++;
    }
```

Typing Game

- 영어 타자 게임

```
        else {
            cout << "오타! 다시 도전!\n";
        }
    }
    end = clock(); //종료
    elapsedTime = (double)(end - start) / CLOCKS_PER_SEC;
    cout << fixed;
    cout.precision(2);
    cout << "게임 소요 시간: " << elapsedTime << "초" << endl;

    return 0;
}
```

문자열 처리 함수

- 문자열 처리 함수 - <string> 헤더파일 포함

함수의 원형	기능 설명
length()	문자열의 개수 반환
at(idx)	인덱스에 있는 문자 반환
substr(pos, len)	부분 문자열 반환
replace(pos, len, str)	일부 문자열 대체
find(str)	문자열 찾아서 위치 반환, 못찾으면 -1
compare()	문자열 비교 – 일치(0), 대소 비교

문자열 처리 함수

- 문자열 처리 함수 - <string> 헤더파일 포함

```
string str = "Hello World";

//문자열의 길이
cout << str.length() << endl;  // 11

//문자열 인덱싱
cout << str.at(0) << endl;     // H
cout << str[6] << endl;        // W

string jumin = "030815-4567890";
char gender = jumin.at(7); //4

switch (gender) {
case '1': case '3':
    cout << "남자" << endl;
    break;
case '2': case '4':
    cout << "여자" << endl;
    break;
}
```

문자열 처리 함수

- 문자열 처리 함수 - <string> 헤더파일 포함

```
// 문자열 자르기(추출)
string sub = str.substr(0, 5);  // "Hello"
cout << sub << endl;

// 대체
str.replace(6, 5, "Korea");
cout << str << endl;          // "Hello Korea"

// 찾기
int pos = str.find("Korea");  //찾으면 위치 반환 6
//int pos = str.find("Java"); //찾지 못하면 -1
cout << pos << endl;

// 비교
string a = "apple", b = "banana";
if (a.compare(b) < 0)
    cout << "a가 b보다 작음" << endl;
if (a.compare(b) == 0)
    cout << "a와 b는 같은 문자열임" << endl;
```

실습 문제 – 문자열 함수

2025년 민생회복 지원금 신청 프로그램을 아래와 같이 구현하세요.

[파일이름: coupon.cpp]

	월	화	수	목	금
출생연도 마지막 자리	1	2	3	4	5
	6	7	8	9	0

☞ 실행 결과

```
출생연도 입력: 2026
신청일은 월요일입니다.
```

```
출생연도 입력: 200
입력 오류: 출생년도는 4자리여야 합니다.
```

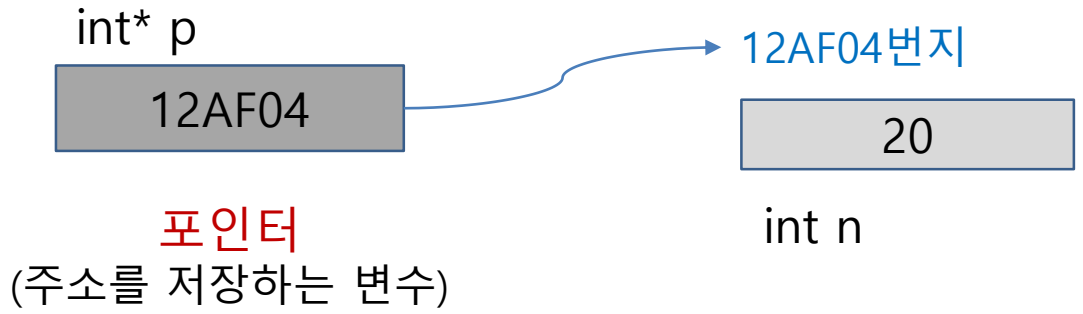
포인터(Pointer)

- 포인터란?

  모든 메모리는 주소(address)를 갖는다. 이러한 **메모리 주소를 저장**하기 위해 사용되는 변수를 포인터 변수라 한다.

  포인터 변수를 선언할 때에는 데이터 유형과 함께 '*' 기호를 써서 나타낸다.

  (예) 택배 주소만 있으면 집을 찾을 수 있다.



동적 메모리 할당

- C와 C++ 동적 메모리 할당 비교

구분	C언어	C++
메모리 생성	malloc()	**new**
메모리 해제	free()	**delete / delete[ ]**
사용 예	int* p = (int*)malloc(sizeof(int) * 10)	**int* ptr = new int[10]**

동적 메모리 할당과 해제

- 정수형 포인터 동적 할당

```
//정적 포인터 변수
int n = 3;
int* p;

p = &n;

cout << n << endl;  //3
cout << *p << endl; //3
cout << *p + 10 << endl; //13
cout << "===============\n";
```

동적 메모리 할당과 해제

- 정수형 포인터 동적 할당

```
//동적 포인터 변수 - new 키워드 사용
int* ptr;
ptr = new int;
if (ptr == nullptr) { //C언어 - NULL, C++ - nullptr
    cout << "메모리를 할당할 수 없습니다.\n";
    return -1;
}

*ptr = 5;
cout << *ptr << endl;  //5
cout << *ptr + 5 << endl;  //6

delete ptr;  //메모리 해제
```

동적 메모리 할당과 해제

- 정수형 배열 동적 할당

```
//정수형 배열 동적 할당
int* pa;
pa = new int[10];

//값 할당
pa[0] = 1;
pa[1] = 2;
pa[2] = 3;

//값 출력
cout << pa[0] << " " << pa[1] << " " << pa[2] << endl;
cout << *pa << " " << *(pa + 1) << " " << *(pa + 2) << endl;
```

동적 메모리 할당과 해제

- 정수형 배열 동적 할당

```
//1부터 10까지 저장
for (int i = 0; i < 10; i++) {
    pa[i] = i + 1;
    //*(pa + i) = i + 1;
}

//1부터 10까지 출력
for (int i = 0; i < 10; i++) {
    cout << pa[i] << " ";
    //cout << *(pa + i) << " ";
}

delete[] pa;  //메모리 해제(반납)
```

```
1 2 3
1 2 3
1 2 3 4 5 6 7 8 9 10
```

delete[ ] 포인터 // 배열로 할당된 메모리 해제

동적 메모리 할당과 해제

- 동적 포인터 배열의 연산

```
int num;      //배열의 개수
int sum = 0; //총점
float avg;    //평균
int* score;  //점수를 저장할 포인터

cout << "*** 점수의 평균 계산 프로그램 ***\n";
cout << "입력할 정수의 개수: ";
cin >> num;

score = new int[num]; //배열의 동적 할당
if (score == nullptr) {
    cout << "메모리를 할당할 수 없습니다.\n";
    return -1;
}
```

동적 메모리 할당과 해제

- 동적 포인터 배열의 연산

```
//점수 입력 및 총점 계산
for (int i = 0; i < num; i++) {
    cout << i + 1 << "번째 점수: ";
    cin >> score[i];

    sum += score[i];
}

//평균
avg = (float)sum / num;

cout << fixed;
cout.precision(1);
cout << "평균: " << avg << endl;

delete[] score;  //메모리 해제
```

```
*** 점수의 평균 계산 프로그램 ***
입력할 정수의 개수: 3
1번째 점수: 70
2번째 점수: 85
3번째 점수: 90
평균: 81.7
```

성적 통계 분석

- 성적 통계 분석

```
-----------------------------------------------------
1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
-----------------------------------------------------
선택> 1
학생수: 3
-----------------------------------------------------
1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
-----------------------------------------------------
선택> 2
score[0]=70
score[1]=85
score[2]=90
-----------------------------------------------------
1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
-----------------------------------------------------
선택> 3
score[0]=70
score[1]=85
score[2]=90
-----------------------------------------------------
1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
-----------------------------------------------------
선택> 4
평균 점수: 81.7
최고 점수: 90
-----------------------------------------------------
1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료
-----------------------------------------------------
선택> 5
프로그램 종료!
```

성적 통계 분석

- 성적 통계 분석

```
int main()
{
    bool run = true;  //상태 변수
    int choice;       //메뉴
    int studentNum;   //학생 수
    int* score = nullptr;  //점수(동적 배열)

    while (run) {
        cout << "-----------------------------------------------------\n";
        cout << "1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료\n" ;
        cout << "-----------------------------------------------------\n";
        cout << "선택> ";

        cin >> choice;  //메뉴 선택
```

성적 통계 분석

- 성적 통계 분석

```
switch (choice) {
case 1:
    cout << "학생수 입력: ";
    cin >> studentNum;
    score = new int[studentNum];
    if (score == nullptr) {
        cout << "메모리를 할당할 수 없습니다.\n";
        return 1;
    }
    break;
case 2:
    for (int i = 0; i < studentNum; i++) {
        cout << "score[" << i << "]=";
        cin >> score[i];
    }
    break;
```

성적 통계 분석

- 성적 통계 분석

```
case 3:
    for (int i = 0; i < studentNum; i++) {
        cout << "score[" << i << "]=" << score[i] << endl;
    }
    break;
case 4:
    //평균점수, 최고점수
    int total, max;   //총점, 최고점수
    float average;    //평균

    total = 0;
    max = score[0];     //최대값 설정
    for (int i = 0; i < studentNum; i++) {
        total += score[i];  //총점

        if (score[i] > max) //최고점수
            max = score[i];
    }
    average = (float)total / studentNum;
```

성적 분석

- 성적 통계 분석

```
        case 5:
            cout << "프로그램 종료!\n";
            run = false;
            break;
        default:
            cout << "잘못된 선택입니다. 다시 선택하세요\n";
            break;
        }
}//while() 닫음

delete[] score;

return 0;
```

성적 분석 – 함수 사용

- 전역변수 선언, 학생수

```
//전역 변수 선언
bool run = true;  //상태 변수
int studentNum;   //학생 수
int* score;  //점수(동적 배열)

void getStudentNum() { //학생수
    if (score != nullptr) {
        delete[] score;  //메모리 제거
        score = nullptr;
    }

    cout << "학생수 입력: ";
    cin >> studentNum;
    score = new int[studentNum];
    if (score == nullptr) {
        cout << "메모리를 할당할 수 없습니다.\n";
        return;
    }
}
```

성적 분석 – 함수 사용

- 점수 입력 / 점수 리스트

```
void scoreNullptr() { //2, 3, 4 메뉴를 먼저 선택한 경우
    if (score == nullptr) {
        puts("먼저 학생수를 입력하세요.");
        return;
    }
}

void inputScore() { //점수 입력
    scoreNullptr();

    for (int i = 0; i < studentNum; i++) {
        cout << "score[" << i << "]=";
        cin >> score[i];
    }
}

void getScoreList() { //점수 리스트
    scoreNullptr();

    for (int i = 0; i < studentNum; i++) {
        cout << "score[" << i << "]=" << score[i] << endl;
    }
}
```

성적 분석 – 함수 사용

- 통계 분석

```
void calculate() { //분석 통계
    scoreNullptr();

    int total = 0; //총점
    int max = score[0]; //최대값 설정
    float average; //평균

    for (int i = 0; i < studentNum; i++) {
        total += score[i];  //총점

        if (score[i] > max) //최고점수 비교
            max = score[i];
    }
    average = (float)total / studentNum;

    cout << fixed;  //소수 자리수 설정
    cout.precision(1);
    cout << "평균 점수: " << average << endl;
    cout << "최고 점수: " << max << endl;
}
```

성적 분석 – 함수 사용

- 메인 함수

```
int main()
{
    while (run) {
        cout << "-----------------------------------------------------\n";
        cout << "1.학생수 | 2.점수입력 | 3.점수리스트 | 4.분석 | 5.종료\n";
        cout << "-----------------------------------------------------\n";
        cout << "선택> ";

        int choice;       //메뉴 선택
        cin >> choice;  //메뉴 선택

        switch (choice) {
        case 1:
            getStudentNum();
            break;
        case 2:
            inputScore();
            break;
```

성적 분석 – 함수 사용

- 메인 함수

```
        case 3:
            getScoreList();
            break;
        case 4:
            calculate();
            break;
        case 5:
            cout << "프로그램 종료!\n";
            run = false;
            break;
        default:
            cout << "잘못된 선택입니다. 다시 선택하세요\n";
            break;
        }
    }

    return 0;
}
```