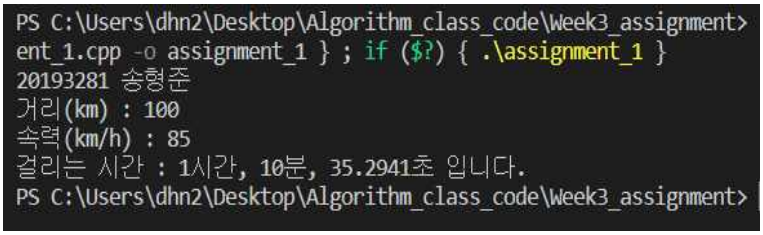
자료구조와 알고리즘 보고서

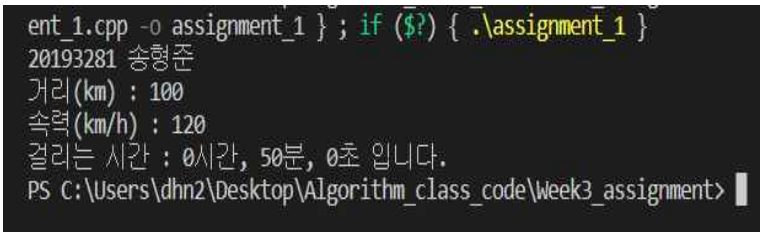
20193281 송형준

1. 거리 및 속력 계산

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

void CalculateTime(double, double);

int main(){

    cout << "20193281 송형준" << endl;

    double distance, speed;

    cout << "거리(km) : ";
    cin >> distance;

    cout << "속력(km/h) : ";
    cin >> speed;

    CalculateTime(distance, speed);

    return 0;
}

void CalculateTime(double dist, double spd){ //거리와 속력을 입력받아, 시간,분,초를 출력

    int hour = dist/spd;
    int minute = (dist/spd - hour)*60;
    double second = ((dist/spd - hour)*60 - minute)*60;

    cout << "걸리는 시간 : " << hour << "시간, " << minute << "분, " << second << "초 입니다." << endl;
}
```

실행 결과





연구조사

목적 : 예상 시간 몇 시간, 몇 분, 몇 초 -> 시간=속력/거리

1. 거리(km), 속력(km/h)를 실수로 입력받는다. -> cin
2. CalculateTime()
   - 시간 = 속력 / 거리 (단위 : 시간) ex) 결과 0.1 = 6분
   - 시 : 거리/속력의 정수부
   - 분 : (거리/속력의 소수부) * 60 의 정수부
   - 초 : ((거리/속력의 소수부) * 60)의 소수부 * 60
3. cout 으로 출력

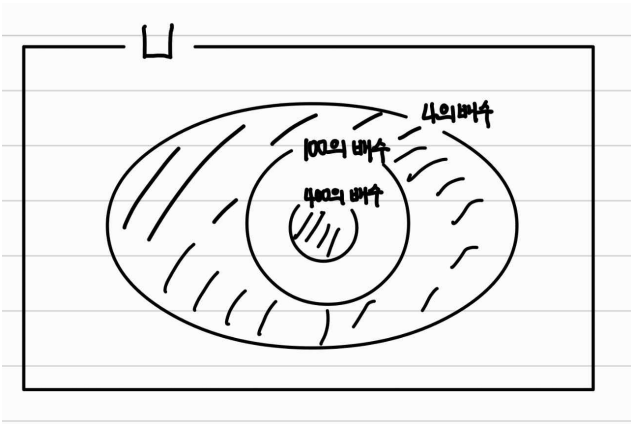
2. 윤년 구하기

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

void is_LeapYear(int year); //윤년 판단 함수

int main(){

    cout << "20193281 송형준" << endl;

    int year;

    cout << "년도 입력 : ";
    cin >> year;

    is_LeapYear(year);

    return 0;
}

void is_LeapYear(int year){

    if(year%400==0 || (year%4==0 && year%100!=0)){
        cout << year <<"는 윤년입니다. " << endl;
    }
    else{
        cout << year <<"는 평년입니다. " << endl;
    }
}
```

실행 결과

```
문제  출력  디버그 콘솔  터미널
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment> cd "c:\U
ent_2.cpp -o assignment_2 } ; if ($?) { .\assignment_2 }
20193281 송형준
년도 입력 : 2022
2022년은 평년입니다.
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
문제  출력  디버그 콘솔  터미널
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment> cd "c:\Use
ent_2.cpp -o assignment_2 } ; if ($?) { .\assignment_2 }
20193281 송형준
년도 입력 : 2020
2020년은 윤년입니다.
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```



연구조사

목적 : 윤년 여부 판단 후 출력

윤년의 조건
1) 4로 나누어 떨어지기
2) 그중 100으로 나누어 떨어지면 평년
3) 400으로 나누어떨어지면 윤년
-> 조건 정리 : 400의 배수이거나, 4의 배수이지만 100의 배수는 아닌 수 (벤다이어그램 참조)

1. 년도를 입력받는다.
2. is_LeapYear()
year%400==0 || year%4==0 && year%100!=0 : 400의 배수이거나, 4의 배수이지만 100의 배수는 아닌 수
3. 출력

3. 숫자들의 개수와 합을 출력

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

void numCalculate(int);

int main(){
    cout << "20193281 송형준" << endl;

    int x;

    cout << "임의의 정수 : ";
    cin >> x;

    if(x<0) x*=-1; //음수일 경우 양수로

    numCalculate(x);
}

void numCalculate(int x){

    int count=0;
    int sum =0;

    while(x){
        int num = x%10; //10으로 계속 나눔 -> 각 자릿수
        x/=10;
        sum+=num;
        count++; //자릿수 판단
    }
    cout << "정수의 자릿수 : " << count << endl;
    cout << "각 자리의 합 : " << sum << endl;
}
```

실행 결과

```
20193281 송형준
임의의 정수 : 12345
정수의 자릿수 : 5
각 자리의 합 : 15
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
20193281 송형준
임의의 정수 : -12345
정수의 자릿수 : 5
각 자리의 합 : 15
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

연구조사

목적 : 정수의 자릿수 판단, 각 자리 숫자의 합

1. 사용자에게 정수 x를 입력받는다.
2. numCalculate(int x)
    - x를 10으로 나누기를 반복한다.
    - x%10 (x의 일의 자리 숫자)를 sum에 더한다.
    - 반복 한번 당 x = x/10 로 재정의한다.
    - 반복 한번 당 count++로 자릿수를 센다.
    - x = 0 일시 반복문 종료

 * 그림 참조

 3. 출력

ex)
x : 1234 -> 123 -> 12 -> 1 -> 0 (반복문 종료)
num : 4 -> 3 -> 2 -> 1 -> 0
sum : 4 -> 7 -> 9 ->10 -> 10 (총합 10)
count : 1 -> 2 -> 3 -> 4 -> 4 (네자리수)

4. 직각삼각형 출력

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

int main(){

    cout << endl << "20193281 송형준" << "₩n";
    cout << "₩n";

    int n;
    cout << "임의의 정수 :(1~9) : ";
    cin >> n;

    if(n<1 || n>9){
        cout << "Out of Range! " << endl;
        return 0;
    }

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(j<=i) cout << j+1;
            else cout << "*";
        }
        cout << endl;
    }
    return 0;
}
```

실행 결과

```
20193281 송형준

임의의 정수 :(1~9) : 8
1*******
12******
123*****
1234****
12345***
123456**
1234567*
12345678
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

연구조사

1. 정수 입력받기(1~9) : 이외의 범위 예외처리
2. 이중 for문
    2-1 for문_i : 줄바꿈용 & 숫자로 출력할 경계선 역할
    2-2 for문_j : j <= i 까지는 숫자 출력, 이후로는 * 출력

5. 이진수를 십진수로 변환하는 함수

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

bool validateBinary(int num);
int binaryToDecimal(int num);

int main(){
    int num;

    cout << endl << "20193281 송형준" << "₩n" ;
    cout << "₩n";

    while(true){ //틀릴시 재입력 받기
        cout << "이진수(1 또는 0): ";
        cin >> num;
        cout << "입력된 이진수 : " << num << endl;

        if(validateBinary(num)){ //if문 실행시 반복문 종료
            num = binaryToDecimal(num);
            break;
        }
        else{
            cout << "이진수(1 또는 0)만 입력 가능합니다!!!" << endl;
        }
    }
    cout << "변환된 십진수 : " << num << endl;
}

bool validateBinary(int num){
    string str = std::to_string(num); //str변환

    for(int i=0;i<str.size();i++){
        if(str[i]< '0' || str[i]> '1'){
            return false;
        }
    }
    return true;
}

int binaryToDecimal(int num){
    string str = std::to_string(num);//str변환

    int sum=0;
```

실행 결과

```
20193281 송형준

이진수(1 또는 0): 1010
입력된 이진수 : 1010
변환된 십진수 : 10
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
20193281 송형준

이진수(1 또는 0): 1234
입력된 이진수 : 1234
이진수(1 또는 0)만 입력 가능합니다!!!
이진수(1 또는 0): 1111
입력된 이진수 : 1111
변환된 십진수 : 15
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
    for(int i=0;i<str.size()-1;i++){
        int BinaryToDec=str[i]-'0';//정수로 변환
        for(int j=0;j<str.size()-(i+1);j++){ //2를 제곱
            BinaryToDec*=2;
        }
        sum+=BinaryToDec;
    }
    sum+=(str[str.size()-1]-'0'); //마지막 자리

    return sum;
}
```

연구조사

1. 이진수 입력
2. bool validateBinary(int num)
    2-1 입력 받은 정수 num을 std::to_string을 이용하여 문자열로 변환한다.
    2-2 for문을 사용하여, 각 index의 값이 0 or 1인지 확인한다.
    2-3 모든 값이 0 or 1일 경우 true, 아니면 false를 반환한다.
3-1. validateBinary()가 반환값이 true라면 binaryToDecimal(int num) 실행
    int binaryToDecimal(int num)
        3-1-1 입력 받은 정수 num을 std::to_string을 이용하여 문자열로 변환한다.
        3-2-2 이중 for문
            for문_i : num의 큰 자릿수부터 접근 & 거듭제곱된 수의 총합 계산
            for문_j : 2의 거듭제곱용 for문, 문자열 size - (i+1) 만큼 거듭제곱해준다.
                ex) 1111에서 i=0일 때 해야 할 거듭제곱 횟수 : 3번 = 4-(0+1)
        3-3-3 마지막 자리수를 sum에 더한다.
3-2. validateBinary()의 반환값이 false라면 재입력을 받는다.

*** 여러 가지 정수 -> 문자열 변환법 ***
(1) std::to_string 기능 사용
(2) std::stringstream 기능 사용
(3) boost::lexical_cast 기능 사용

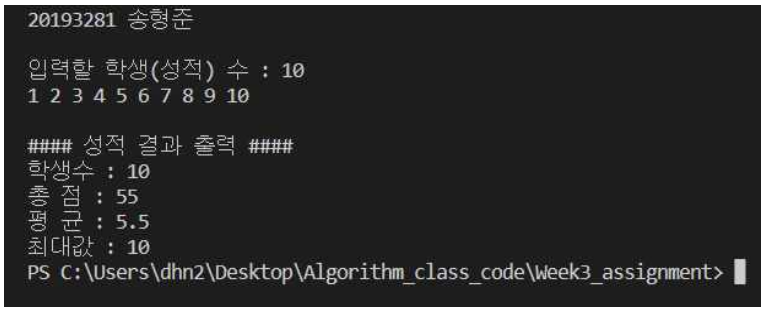
C스타일
atoi & strtoi 사용 --> #include <cstdlib> 후 사용

*** 문자열 -> 정수 변환법 ***
stoi 는 string에만 사용가능, char에는 사용할 수 없음
사용하는 편법 : 아스키 코드를 활용한 str -'0'

6. 학생 성적 처리

```
#include <iostream>
using namespace std;
void SCORE(const double* pArr, int num, double& rSum, double& rAve, double& rMax);

int main(){
    int num;

    cout << endl << "20193281 송형준" << "₩n" ;
    cout << "₩n";

    cout << "입력할 학생(성적) 수 : ";
    cin >> num;

    double* scoreArr = new double[num];

    for(int i=0;i<num;i++){
        cin >> *(scoreArr+i);
    } //num만큼 cin 받기

    cout << "₩n";
    cout << "#### 성적 결과 출력 ####" << endl;

    double sum, average, max;
    SCORE(scoreArr, num, sum, average, max);

    return 0;
}

void SCORE(const double* pArr, int num, double& rSum, double& rAve, double& rMax){
    rMax = *pArr;

    for(int i=0;i<num;i++){
        rSum+=*(pArr+i); //총합구하기

        if(rMax < *(pArr+i)) rMax = *(pArr+i);//최댓값 찾기
    }
    rAve = rSum/(double)num;

    cout << "학생수 : " << num << endl;
    cout << "총 점 : " << rSum <<  endl;
    cout << "평 균 : " << rAve << endl;
    cout << "최대값 : " << rMax <<  endl;
}
```

실행 결과



연구조사

1. 학생수를 입력 받는다.
2. new를 이용해 학생수만큼 메모리를 동적할당 받는다.
3. void SCORE(const double* pArr, int num, double& rSum, double& rAve, double& rMax)
    3-1 for문
        총합 구하기 : rSum+=*(pArr+i)
        최댓값 찾기 : 첫 번째 요소를 최댓값이라 가정, 이보다 더 큰 값이 있을시 해당 index의 값으로 변경
        평균 : rSum/num
4. 출력

double* scoreArr = new double[num] 의 의미
double형 포인터 scoreArr = 가리키는 녀석의 자료형이 double
포인터에 자료형을 부여해야하는 이유 : 가리키는 녀석에 따라 부여할 byte수를 결정하기 때문에.

new : 동적할당 명령어
-> new double[num] : 동적할당할건데, 안에 저장할 자료형은 double이고, 배열 크기는 num이야. 라는 의미
ex) new int; : 동적할당할건데, 안에 저장할 자료형은 int야

7. 클래스 설계: 좌표 값 설정

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

class Point{
    int x;
    int y;
public :
    void setX(int x){
        this->x=x;
    };
    void setY(int y){
        this->y=y;
    };
    int getX(){
        return x;
    };
    int getY(){
        return y;
    };
    void showData(){
        cout << " x 좌표 : " << getX() << endl;
        cout << " y 좌표 : " << getY() << endl;
    };
};

int main(){
    Point pt;

    cout << endl << "20193281 송형준" << "₩n" ;
    cout << "₩n";

    int temp_x, temp_y;
    cout << "temp.x : ";
    cin >> temp_x;
    cout << "temp.y : ";
    cin >> temp_y;

    pt.setX(temp_x);
    pt.setY(temp_y);

    pt.showData();

    return 0;
}
```

실행 결과

```
20193281 송형준

temp.x : 100
temp.y : 200
 x 좌표 : 100
 y 좌표 : 200
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

연구조사

1. temp_x, temp_y에 각 좌표값을 입력받는다.
2. setX, setY로 클래스 내 private 변수 값을 설정한다.
3. showData() 내 getX, getY로 클래스 내 private 변수에 접근, 출력한다.

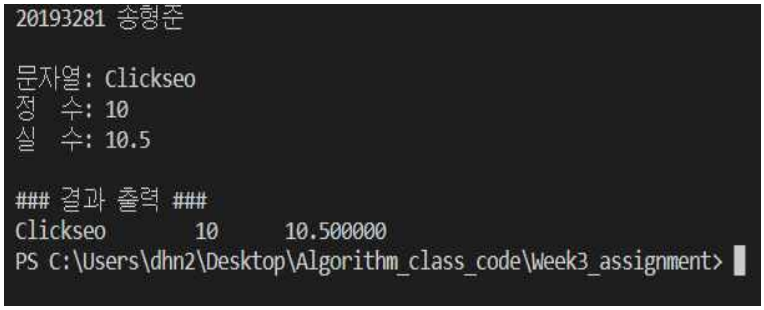
set, get의 필요성 : private 변수에 접근하기 위해서

this 포인터
멤버함수의 인자 원형 : (반환형) 함수이름((객체의 주소값), 인수) ex) void setX(&Point, int x)
this는 객체의 주소값을 저장하는 변수, 즉 객체의 포인터
this->x 는 객체 내의 x를 지칭

인라인함수
함수 호출 과정에서 드는 시간 존재 -> 반복적일시 성능 저하 발생
코드가 짧은 함수는, 함수 정의를 따로하지않고, inline 함수화한다.
class 내에서 함수 내용을 정의하면, 자동으로 inline 함수화된다.

8. cin, cout 객체 구현

실행 결과




```
#define _CRT_SECURE_NO_WARNINGS

#include <iostream>
#include <cstdio>

namespace mystd{
    const char* endl = "₩n";

    class ostream{
    public:
        ostream& operator<<(char* str){ //문자열 출력
            printf("%s", str);
            return *this;
        }

        ostream& operator<<(const char* c){ //endl 출력용
            printf("%s", c);
            return *this;
        }

        ostream& operator<<(int& n){ //정수 출력
            printf("%d", n);
            return *this;
        }

        ostream& operator<<(double& d){ //실수 출력
            printf("%lf", d);
            return *this;
        }
    };

    class istream{
    public:
        istream& operator>>(char* str){
            scanf("%s", str, 1024);
            return *this;
        }

        istream& operator>>(int& n){
            scanf("%d", &n);
            return *this;
        }
        istream& operator>>(double& d){
            scanf("%lf", &d);
            return *this;
```

```
        }
    };
    ostream cout;
    istream cin;
}

using namespace mystd;

int main(){
    char str[100];
    int n;
    double d;

    cout << "문자열: "; //객체 선언 후 << 연산 수행
    cin >> str;

    cout << "정  수: ";
    cin >> n;

    cout << "실  수: ";
    cin >> d;

    cout << endl;
    cout << "### 결과 출력 ###" << endl;
    cout << str << "₩t" <<  n << "₩t" << d;

    return 0;
}
```

연구조사

1. mystd namespace를 정의한다.
2. namespace 내 class ostream, istream을 정의한다.
3. class의 내부함수로 operator <<, >> 함수를 정의한다.
4. mystd 내에 ostream 객체 cout, istream 객체 cin을 생성한다.
4. 만든 mystd::cin객체와 >>연산자를 이용해 사용자에게 문자열, 정수, 실수를 입력 받는다.
5. 만든 mystd::cout객체와 <<연산자를 이용해 값을 출력한다.

scanf의 오류를 예방하고자 #define _CRT_SECURE_NO_WARNINGS을 사용한다.

연산자 다중정의 : 피연산자에 따라 연산자의 기능을 달리 하는 것
(반환형) operator (다중정의하고자 하는 연산자)(매개변수)

<< >> 다중정의
1. 반환형 : 객체의 참조값 : 연산을 수행하는 객체를 참조로 반환 = 객체가 저장된 공간을 반환
    - 객체 반환해야하는 이유?
        << , >> 연산자를 여러번 사용하는 경우를 생각해보자.

ex) cout << 1 << 2 << 3 << 4 << endl;

<< 연산자가 사용되는 순서를 표시하면 아래와 같다.

1. (cout << 1) << 2 << 3 << 4 << endl;
2. (cout << 2) << 3 << 4 << endl;
3. (cout << 3) << 4 << endl;
4. .....

코드가 위와 같이 작동하기 위해서는, cout << 1 의 변환값으로서 cout이 존재해야한다. 즉, 객체 반환값이 존재해야한다.
단순 객체 반환시, call by value 에 의한 얕은 복사, 반복적인 복사로 인한 성능 저하가 발생한다. 따라서 객체 반환을 이용한다.

2. 인수
   -cout : char*, const char*, int, double
   -cin : char*, int, double

3. 반환시 *this : class 자신을 반환 -> cout 객체의 반복적인 사용을 위해

9. 은행 계좌 출금

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

const string client_account="100-1234-5678";
const string clinet_pass="12345";
int balance = 100000; //잔액

class AccountClass{
    string account;
    string pass;
public :
    AccountClass(string account, string pass);
//생성자로 accout, pass 대입
    void showAccount();
};

int main(){

    cout << endl << "20193281 송형준" << "₩n" ;
    cout << "₩n";

    string account;
    string pass;

    cout << " 계좌번호 : ";
    cin >> account;
    cout << " 비밀번호 : ";
    cin >> pass;

    AccountClass AC(account,pass); //입력값으로 객체 생성

    if(account.compare(client_account)==0 && pass.compare(clinet_pass)==0){
        int withdraw = 0;

        cout << " 출금액 : ";
        cin >> withdraw;

        if(withdraw > balance){
            cout << "출금 희망액: (" << withdraw << ")" << " 이 너무 많네요. " << endl;
        }
        else{
            cout << "출금 : " << withdraw << " 잔액 : " << balance - withdraw << endl;
        }
    }
```

실행 결과

```
20193281 송형준

 계좌번호 : 100-1234-5678
 비밀번호 : 12345
 출금액 : 10000
출금 : 10000 잔액 : 90000
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
20193281 송형준

 계좌번호 : 100-1234-5678
 비밀번호 : 12345
 출금액 : 50000000
출금 희망액: (50000000) 이 너무 많네요.
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
20193281 송형준

 계좌번호 : 100-1234-5678
 비밀번호 : 11111
 다음 입력을 다시 한번 확인하세요!!!
 계좌번호 : 100-1234-5678
 비밀번호 : 11111
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

```
    else{
        AC.showAccount();
        return 0;
    }

    return 0;
}

AccountClass::AccountClass(string account, string pass){//생성자로 accout, pass 대입
    this->account = account;
    this->pass = pass;
}

void AccountClass::showAccount(){
    cout << " 다음 입력을 다시 한번 확인하세요!!! " << endl;
    cout << " 계좌번호 : " << this->account << endl;
    cout << " 비밀번호 : " << this->pass << endl;
}
```

연구조사

1. 전역변수로 정답에 해당하는 계좌번호 client_account와 비밀번호 clinet_pass를 선언한다.
2. 전역변수로 잔액을 선언한다.
3. class AccountClass
    account : 사용자에게 입력받은 account를 저장할 변수
    pass : 사용자에게 입력받은 pass를 저장할 변수
    AccountClass(string account, string pass) : account, pass 값을 입력받는 생성자
    showAccount() : 클래스 내 account, pass를 출력
4. 계좌번호, 비밀번호를 입력받는다.
5. 입력받은 계좌번호, 비밀번호로 AccountClass 객체를 생성한다.
6. string 내 메소드함수 compare()를 사용하여, client_account, clinet_pass와 비교한다.

7-1. 일치할 경우, 출금액을 입력받고, 잔액과 비교한다
    7-1-1 출금액 > 잔액 : 출금 불가능 문구 출력
    7-1-2 출금액 < 잔액 : 출금 금액과 남은 잔액 출력

7-2 불일치할 경우, AccountClass::showAccount() 실행 후 코드 종료
    AccountClass::showAccount() : 객체 내의 account, pass 출력

10. 함수 템플릿

```
#include <iostream>

using namespace std;

template <class T>
void printArr(const T* pArr,const int num){

    for(int i=0;i<num;i++){
        cout << *(pArr+i) << " ";
    }
    cout << endl;
}

int main(){
    cout << endl << "20193281 송형준" << "₩n" ;
    cout << "₩n";

    char str[]="Hi ~ Clickseo";
    int iArr[]={10,20,30,40,50};
    double dArr[]={10.5,20.5,30.5,40.5,50.5};

    printArr(str,sizeof(str)/sizeof(char));
    printArr(iArr,sizeof(iArr)/sizeof(int));
    printArr(dArr,sizeof(dArr)/sizeof(double));

    return 0;
}
```

실행 결과

```
20193281 송형준

Hi ~ Clickseo
10 20 30 40 50
10.5 20.5 30.5 40.5 50.5
PS C:\Users\dhn2\Desktop\Algorithm_class_code\Week3_assignment>
```

연구조사

1. 문자열, 정수배열, 실수배열을 선언한다.
2. printArr를 통해 출력한다.

함수 템플릿 : 자료형에 관계없이 일반화된 함수를 작성할 수 있도록 해주는 도구

template <class or typename (이름)>
반환형 함수이름(매개변수 1, ...)

<class or typename (이름)> : 제네릭타입이라 부르며, 자료형의 일반화이다, 실제 사용시 원하는 자료형을 부여할 수 있다. 하나의 제네릭타입 당 하나의 자료형을 부여할 수 있다.

ex) template <class T1, class T2> void func(T1 a, T2b) 의 경우, T1에 하나의 자료형, T2에 하나의 자료형을 부여할 수 있다.
func(3 , 90.1234) 의 경우 T1에는 int형, T2에는 double형이 부여된다.

문제의 T 자료형은 아래와 같다.

```
printArr(str,sizeof(str)/sizeof(char)) -> T의 자료형 : 문자
printArr(iArr,sizeof(iArr)/sizeof(int)); -> T의 자료형 : 정수
printArr(dArr,sizeof(dArr)/sizeof(double)); -> T의 자료형 : 실수
```

sizeof() : 입력받은 값의 자료형에 따른 바이트수를 반환한다.
배열의 경우, sizeof(배열이름)을 통해 배열 길이 * 바이트수를 반환 받을 수 있다.
이를 이용해 sizeof(배열이름)/sizeof(자료형) 으로 배열의 길이를 알 수 있다.