dongguk UNIVERSITY

심화프로그래밍

실습강의 4주차

실습강의 소개

- **실습 진행 방법**
  - 간단한 이론 복습 및 해당주차 실습과제 설명
  - 실습 후 보고서와 소스코드를 압축하여 **수요일 자정(23:59)까지**

    꼭!! 이클래스 제출(**이메일 제출 불가, 반드시 이클래스를 통해 제출**)
  - 실습과제 제출기한 엄수(제출기한 이후로는 0점 처리)
- **Q & A**
  - 이클래스 및 실습조교 이메일을 통해 질의응답
  - **이메일 제목 : [심화프로그래밍_홍길동] *본인 과목명과 성명 꼭 작성!!**
  - **실습조교 메일 주소 : 0hae@dgu.ac.kr, wundermilch@dgu.ac.kr**

실습 보고서 작성 방법 [1/2]

- **실습 보고서**
  - **문제 분석: 실습 문제에 대한 요구 사항 파악, 해결 방법 등 기술**
  - **프로그램 설계 및 알고리즘**
    - 해결 방법에 따라 프로그램 설계 및 알고리즘 등 기술
    - e.g.) 문제 해결 과정 및 핵심 알고리즘 기술
  - **소스코드 및 주석**
    - 소스코드와 그에 해당하는 주석 첨부
    - 각각의 함수가 수행하는 작업, 매개변수, 반환 값 등을 명시
    - 소스코드 전체 첨부**(소스코드 화면 캡처X, 소스코드는 복사/붙여넣기로 첨부)**
  - **결과 및 결과 분석**
    - 결과 화면을 캡쳐 하여 첨부, 해당 결과가 도출된 이유와 타당성 분석
  - **소감**
    - 실습 문제를 통해 습득할 수 있었던 지식, 느낀 점 등을 기술

실습 보고서 작성 방법 [2/2]

- **제출 방법**
  - 보고서, 소스코드를 1개의 파일로 압축하여 e-class "과제" 메뉴를 통해 제출
    - **"이름학번실습주차.zip" 형태로 제출(e.g. :김동국19919876실습3.zip)**
    - 파일명에 공백, 특수 문자 등 사용 금지
- **유의 사항**
  - 보고서의 표지에는 학과, 학번, 이름, 담당 교수님, 제출일자 반드시 작성
  - 정해진 기한내 제출
    - 기한 넘기면 **0점** 처리
    - 이클래스가 과제 제출 마지막 날 오류로 동작하지 않을 수 있으므로, 최소 1~2일전에 제출
    - **과제 제출 당일 이클래스 오류로 인한 미제출은 불인정**
  - 소스코드, 보고서를 자신이 작성하지 않은 경우 **실습 전체 점수 0점 처리**
  - **Visual Studio 2019 또는 Sharstra 웹 IDE 기반 학습 프로그램 사용하여 실습 진행**

namespace와 std:: [1/2]

- **namespace**
  - 여러 프로젝트나 여러 사람들이 작성한 프로그램에서 변수, 함수 클래스 등의 이름(identifier)이 충돌하는 것을 막기 위해, 개발자가 자신만의 고유한 이름 공간을 생성할 수 있도록 함

```
1.  namespace myFn { // myFn라는 이름 공간 생성
2.      ............// 이곳에 선언되는 모든 이름은 모두 myVar
3.  }
```

- 이름을 사용하기 위해서는 다음처럼 이름 공간을 함께 사용함
  - 이름공간(namespace) :: 이름(identifier)
- 사용 예시

```
1. #include "youFn"
2.
3. namespace myFn {
4.     int f() {
5.         return 1;
6.     }
7.     void n() {
8.         f();
9.         yourFn::f();
10.    }
11.}
```

```
1. namespace yourFn {
2.     int f() {
3.         return -1;
4.     }
5.     int g() {
6.         return 0;
7.     }
8. }
```

namespace와 std:: [2/2]

- **std::의 생략과 using 지시어**
  - using 지시어를 이용하면 이름 공간 접두어를 생략할 수 있음
  - namespace 키워드와 함께 using 지시어를 사용하면 됨
- 사용 예시 1

```
1. using std::cout; // cout에 대해서만 std:: 생략
2. ...
3. cout << "Hello" << std::endl;
```

- 사용 예시 2

```
1. #include <iostream> // cout과 << 연산자 정의 포함
2. using namespace std;
3.
4. // C++ 프로그램은 main() 함수에서부터 실행을 시작한다.
5. int main() {
6.     cout << "Hello\n"; // 화면에 Hello를 출력하고 다음 줄로 넘어감
7.     cout << "맛보기";
8.     return 0;
9. }
```

C++ 클래스와 객체

- **클래스**
  - "객체를 만들어내기 위해 정의된 설계도 혹은 틀"
  - 객체를 만들어내기 위해 정의된 설계도, 틀
  - 클래스는 객체가 아님. 실체도 아님
  - 멤버 변수와 멤버 함수 선언

- **객체**
  - "클래스의 모양을 가지고 탄생한 것"
  - 멤버 변수와 멤버 함수로 구성
  - 메모리에 생성, 실체(instance)라고도 부름
  - 하나의 클래스 틀에서 찍어낸 여러 개의 객체 생성 가능
  - 객체들은 상호 별도의 공간에 생성

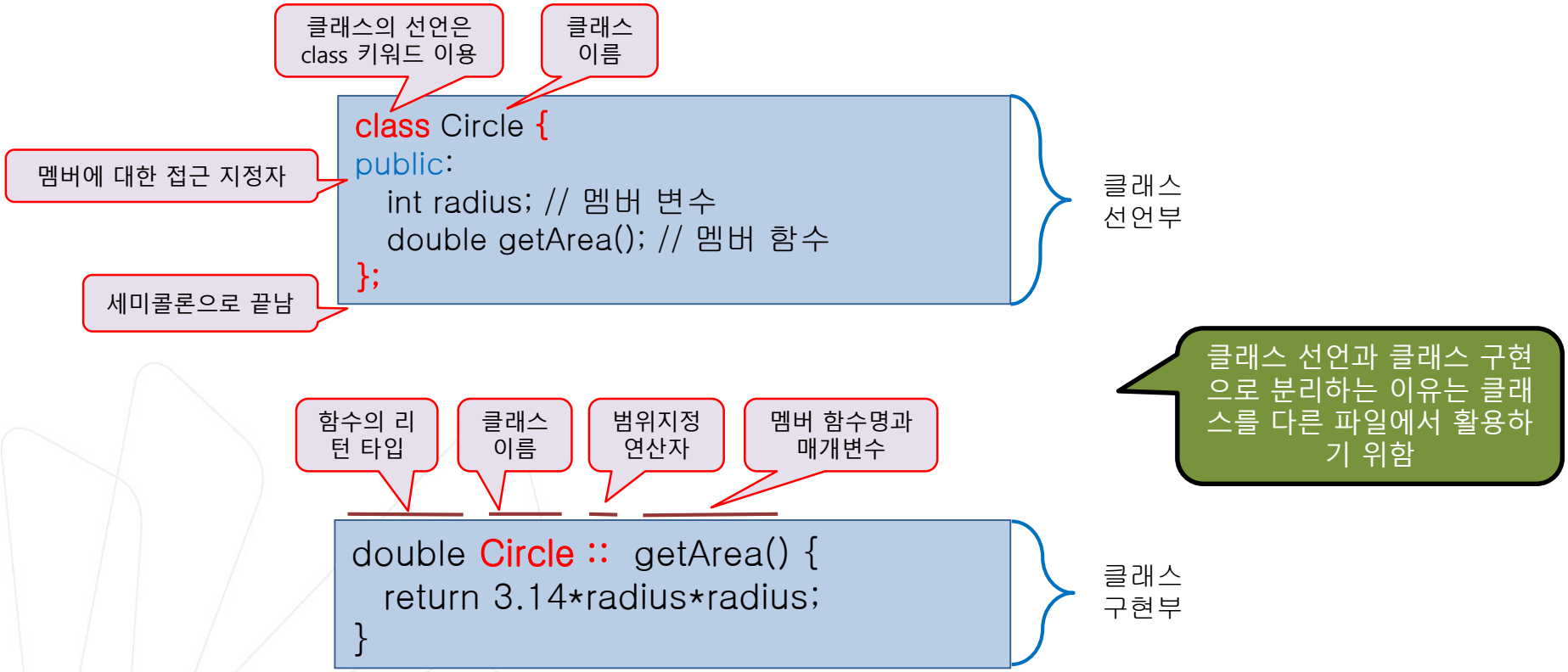
클래스 만들기

- **클래스 작성**
  - 멤버 변수와 멤버 함수로 구성
  - 클래스 선언부와 클래스 구현부로 구성

- **클래스 선언부(class declaration)**
  - class 키워드를 이용하여 클래스 선언
  - 멤버 변수와 멤버 함수 선언
    - 멤버 변수는 클래스 선언 내에서 초기화할 수 없음
    - 멤버 함수는 원형(prototype) 형태로 선언
  - 멤버에 대한 접근 권한 지정
    - private, public, protected(상속 관계일 때 사용) 중의 하나
    - 디폴트는 private(비공개 멤버, 클래스 내에서만 접근 가능)
    - public : 다른 모든 클래스나 객체에서 멤버의 접근이 가능함을 표시

- **클래스 구현부(class implementation)**
  - 클래스에 정의된 모든 멤버 함수 구현

클래스 만들기 설명



실행문 중간에 변수 선언

- **C++의 변수 선언**
  - 변수 선언은 아무 곳이나 가능

```
1.  int width;
2.  cin >> width; //키보드로부터 너비를 읽는다.
3.
4.  cout << "높이를 입력하세요>>";
5.
6.  int height;
7.  cin >> height; //키보드로부터 높이를 읽는다.
8.
9.  //너비와 높이로 구성되는 사각형의 면적을 계산한다.
10. int area = width * height;
11. cout << "면적은 " << area << "\n"; //면적을 출력하고 한 줄 띈다.
```

실행문 중간에 변수 선언

  - C++ 변수 선언 방식의 장점
    - 변수를 사용하기 직전 선언함으로써 변수 이름에 대한 타이핑 오류 줄임
  - C++ 변수 선언 방식의 단점
    - 선언된 변수를 일괄적으로 보기 힘듦
    - 코드 사이에 있는 변수 찾기 어려움

타이핑 오류 가능성 해소

- 선언부에 모든 변수를 선언하는 경우, 타이핑 오류 가능

```
1.  int time, timer;
2.  ...
3.  timer = 5; //time에 5을 저장하려다 timer로 잘못 입력.
4.             //컴파일 오류 발생하지 않음
5.             //그러나 잘못된 실행 결과 발생
6.  ....
7.  timer = 3;
```

- 변수 사용 전에 변수를 선언하면, 타이핑 오류 사전 발견

```
1.  int time;
2.  timer = 5; //time에 5을 저장하려다 timer로 잘못 입력.
3.             //컴파일 오류 발생
4.  ....
5.  int timer;
6.  timer = 3;
```

컴파일 오류

C++의 문자열

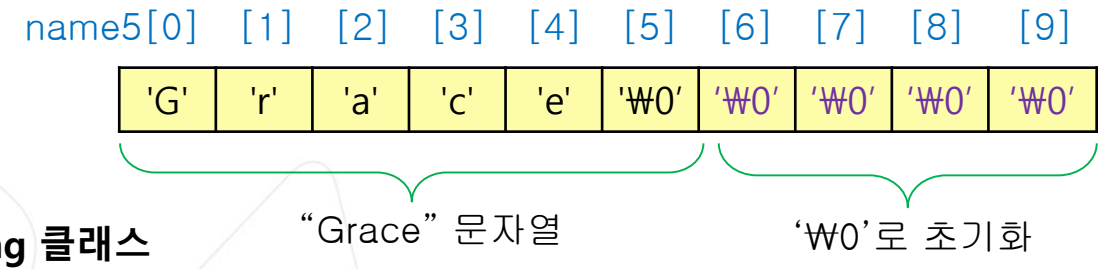
- **C++의 문자열 표현 방식 : 2가지**
  - C-스트링 방식 – '₩0'로 끝나는 문자 배열

```
1.  //name1은 문자열 "Grace"
2.  char name1[6] = {'G', 'r', 'a', 'c', 'e', '\0'};
3.  //name2는 문자열이 아니고 단순 문자 배열
4.  char name2[5] = {'G', 'r', 'a', 'c', 'e'};
```

C-스트링 문자열

단순 문자 배열

```
1.  char name5[10] = "Grace";
```



  - **string 클래스**
    - C++에서 강력 추천, C++ 표준 클래스, 문자열의 크기에 따른 제약 없음
      - string 클래스가 스스로 문자열 크기에 맞게 내부 버퍼 조절
    - 문자열 복사, 비교, 수정 등을 위한 다양한 함수와 연산자 제공
    - <string> 헤더 파일에 선언
      - #include <string> 필요
    - C-스트링보다 다루기 쉬움

C-스트링 방식으로 문자열 다루기

- **C-스트링으로 문자열 다루기**
  - C 언어에서 사용한 함수 사용 가능
    - strcmp(), strlen(), strcpy() 등
  - <cstring>이나 <string.h> 헤더 파일 include
  - <cstring> 헤더 파일을 사용하는 것이 바람직함

```
1. #include <iostream>
2. #include <cstring>
3. // include <string.h>
4. using namespace std;
5.
6. int main() {
7.     char password[11];
8.     cout << "프로그램을 종료하려면 암호를 입력하세요" << endl;
9.     while(true) {
10.        cout << "암호>>";
11.        cin >> password;
12.        if(strcmp(password, "C++") == 0) {
13.            cout << "프로그램 정상 종료" << endl;
14.            break;
15.        }
16.        else
17.            cout << "틀린 암호입니다" << endl;
18.    }
19.}
```

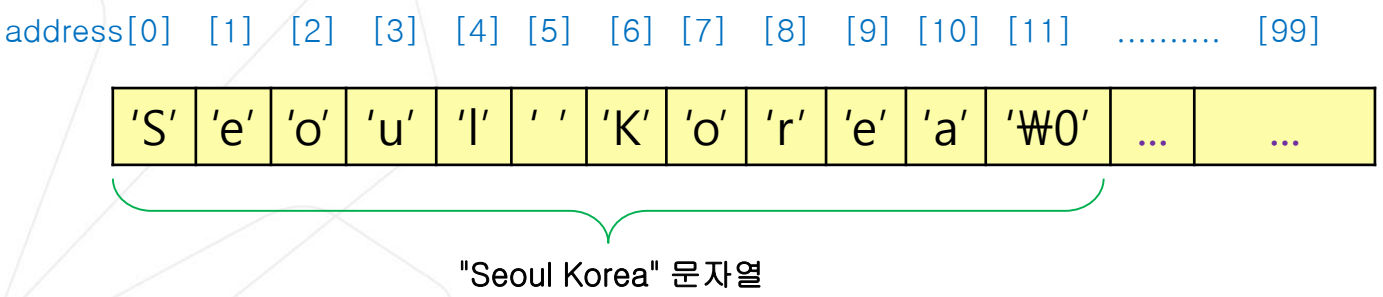
cin.getline()으로 공백이 낀 문자열 입력

- **공백이 낀 문자열**을 입력 받는 방법
  - cin.getline(char buf[], int size, char delimitChar)
    - buf에 최대 size-1개의 문자 입력. 끝에 '₩0' 붙임
    - delimitChar를 만나면 입력 중단. 끝에 '₩0' 붙임
      - delimitChar의 디폴트 값은 '₩n'(<Enter>키)

```
1. char address[100];
2. cin.getline(address, 100, '\n');
```

최대 99개의 문자를 읽어 address 배열에 저장. 도중에 <Enter> 키를 만나면 입력 중단

사용자가 'Seoul Korea<Enter>'를 입력할 때,



클래스와 구조체의 차이 [1/4]

- **C/C++ 구조체**
  - 구조체를 선언하는 방법

```
1.  struct myStruct
2.  {
3.      char a[10];
4.      int b;
5.      float c;
6.  };
```

C언어 구조체 선언

```
1.  typedef struct myStruct
2.  {
3.      char a[10];
4.      int b;
5.      float c;
6.  } typeStruct;
```

C언어 구조체 선언 typedef

```
1.  struct myStruct
2.  {
3.      char a[10];
4.      int b;
5.      float c;
6.  };
```

C++ 구조체 선언

  - C++에서는 typedef를 추가할 필요없이 사용 가능

```
1.  int main()
2.  {
3.      //struct myStruct exam1 = { "Hello", 20, 12.9 };
4.      //typeStruct exam1 = { "Hello", 20, 12.9 };
5.      myStruct exam1 = { "Hello", 20, 12.9 }; // C++ 구조체
6.
        exam1.b = 5023;
7.      exam1.c = 58.45;
8.
9.      cout << exam1.a << " \n" << exam1.b << " \n" << exam1.c << endl;
10.     return 0;
11. }
```


클래스와 구조체의 차이 [2/4]

- **클래스와 구조체**
  - 클래스와 구조체 둘 다 연관 있는 데이터를 묶을 수 있는 문법적 장치
  - 구조체를 클래스로 바꾸고 싶다면 struct를 지우고 class로 변경
    - 프로그램 컴파일 시 오류 발생!!
  - 컴파일 오류가 발생하는 이유는 구조체와 클래스의 접근법의 차이가 있기 때문
- **접근지정자 or 접근제어 지시자 or Access modifier**
  - public : 어디서든 접근 허용
  - protected : 상속 관계에 놓여있을 때, 유도 클래스(자식 클래스)에서의 접근 허용
  - private : 클래스 내(클래스 내에 정의된 함수)에서만 접근 허용
- **구조체의 경우, 모든 변수와 함수는 public으로 선언**
- **클래스의 경우, 접근지정자가 없으면 모든 변수와 함수는 private으로 선언**
  - 다음 장에서 구조체 문법을 클래스 문법으로 변경하는 예제 실습 진행

클래스와 구조체의 차이 [3/4]

- **구조체를 클래스로 변경하기**
  - 구조체(struct)를 클래스(class)로 변경
  - 접근지정자 private, public 사용

접근 지정자 사용
2번 라인, 5번 라인

```
1.  class myClass {
2.  private:
3.      int a;
4.      float b;
5.
6.  public:
7.      void init() {
8.          a = 1;
9.          b = 2.67;
10.         cout << a << b << endl;
11.     }
12. };
13.
14. int main()
15. {
16.     myClass exam2;
17.     exam2.init();
18.     return 0;
19. }
```

멤버 변수, 멤버 함수 선언부

멤버 함수 구현부

멤버 함수 호출

클래스와 구조체의 차이 [4/4]

- **캡슐화(Encapsulation)**
  - 클래스 내부의 속성 및 함수를 외부로부터 감추는 방법
  - 클래스 내부의 멤버 변수나 멤버 함수를 직접적으로 제어할 경우, 예기치 못한 오류나 코드 가독성이 떨어질수 있기 때문에 외부로부터 접근할 수 없도록 private를 사용하고 public에서 제어하는 멤버 함수를 만들어서 사용하는 것이 일반적인 케이스

```
1. class Player {
2. private:
3.     void PlayerDie() {
4.         cout << "Die";
5.         }
6.     int _hp = 100;
7.
8. public:
9.     void SetHP(int hp) {
10.        _hp = hp;
11.        if (_hp <= 0) PlayerDie(); // PlayerDie를 실행하기 위해서는 SetHP를 이용해야만함
12.    }
13.};
```

클래스의 멤버 함수 [1/3]

- **클래스 내부의 멤버 함수**
  - 클래스는 객체를 정의하기 위한 문법으로 내부에 **멤버 변수**와 **멤버 함수**를 선언 가능
  - 간단한 클래스의 경우에는 멤버 함수의 구현부가 다 들어가도 상관 없겠지만,
  - 복잡한 클래스의 경우에는 **상당한 수의 멤버 함수**가 들어갈 것이고 소스코드도 **길다**
  - 이렇게 되면 클래스 내에 어떤 함수들이 있는지 파악하기 어려워져 관리가 힘들다
- **멤버 함수를 클래스 내부에서 외부로**
  - 클래스 내부에는 멤버 함수의 **원형**만 선언
    - void print()
    - void SetRect(int l, int t, int r, int b)
  - **스코프 연산자(::)**를 이용하여 클래스 내부에서 외부로 끌고 나올 수 있다
    - void CRect::print()
    - void CRect::SetRect(int l, int t, int r, int b)

클래스의 멤버 함수 [2/3]

- **멤버 함수 1**
  - 클래스 내부에 멤버 함수 구현하기

```
1. #include <iostream>
2. using namespace std;
3.
4. class CRect {
5.     int left, top, right, bottom;
6. public:
7.     void print() {
8.         cout << "(" << left << "," << top << "," << right << "," << bottom << ")" << endl;
9.     }
10.    void SetRect(int l, int t, int r, int b) {
11.        left = l,
12.        right = r;
13.        top = t;
14.        bottom = b;
15.    }
16.};
17.
18.void main() {
19.    CRect rc;
20.    rc.SetRect(0, 0, 20, 20);
21.    rc.print();
22.}
```

클래스의 멤버 함수 [3/3]

- **멤버 함수 2**
  - 클래스 외부에 멤버 함수 구현하기

```
1. class CRect {
2.     int left, top, right, bottom;
3. public:
4.     void print();
5.     void SetRect(int l, int t, int r, int b);
6. };
7.
8. void CRect::print() {
9.     cout << "(" << left << "," << top << "," << right << "," << bottom << ")" << endl;
10.}
11.
12.void CRect::SetRect(int l, int t, int r, int b) {
13.     left = l,
14.     right = r;
15.     top = t;
16.     bottom = b;
17.}
18.
19.void main() {
20.     CRect rc;
21.     rc.SetRect(0, 0, 20, 20);
22.     rc.print();
23.}
```

생성자

- **생성자(Constructor)**
  - 해당 클래스의 객체가 인스턴스화될 때 **자동으로 호출되는** 특수한 종류의 멤버 함수
  - 생성자는 일반적으로 클래스의 멤버 변수를 적절한 기본값 또는 사용자 제공 값으로 **초기화**하거나 클래스를 사용하는 데 필요한 설정이 필요한 경우 사용
- **생성자 정의 방법**
  - 생성자 이름은 클래스와 이름이 같아야 한다
  - 생성자는 리턴 타입이 없다
- **기본 생성자 (Default constructor)**
  - 매개 변수를 갖지 않거나 모두 기본값이 설정된 매개 변수를 가지고 있는 생성자
- **매개 변수가 있는 생성자를 사용한 초기화**
  - 클래스 인스턴스별 멤버 변수의 값을 특정한 값으로 초기화할 때 사용

기본 생성자 (Default constructor)

- 기본 생성자를 사용한 예제

```
1. class Fraction
2. {
3. private:
4.     int m_numerator;   // 분자
5.     int m_denominator; // 분모
6. public:
7.     Fraction() // 기본 생성자
8.     {
9.          m_numerator = 0;
10.         m_denominator = 1;
11.    }
12.    int getNumerator() { return m_numerator; }
13.    int getDenominator() { return m_denominator; }
14.    double getValue() { return static_cast<double>(m_numerator) / m_denominator; }
15.};
16.
   int main()
17.{
18.    Fraction frac;
19.    cout << frac.getNumerator() << "/" << frac.getDenominator() << '\n';
20.    return 0;
21.}
```

매개 변수가 있는 생성자를 사용한 초기화

- 매개 변수가 있는 생성자를 사용한 예제

```
1. class Date {
2. public:
3.     int year;
4.     int month;
5.     int day;
6.
7.     Date(int year, int month, int day) { // 매개 변수가 있는 생성자
8.         this->year = year;
9.         this->month = month;
10.        this->day = day;
11.        cout << "[Constructor] Date() - param: year, month, day\n";
12.    };
13.    void ShowData() {
14.        cout << "Year: " << year << "\n";
15.        cout << "month: " << month << "\n";
16.        cout << "day: " << day << "\n";
17.    }
18.};
19.
20.int main() {
21.    Date date(2022, 9, 21);
22.    date.ShowData();
23.    return 0;
24.}
```

소멸자 [1/2]

- **소멸자(Destructor)**
  - **객체가 소멸될 때** 자동으로 실행되는 클래스의 멤버 함수
    - 객체가 지역 범위를 벗어나거나 동적으로 할당된 객체가 삭제 키워드를 통해 소멸될 때 실행
  - 생성자는 클래스의 초기화를 돕지만, **소멸자는 일종의 청소를 돕도록 설계**
- **소멸자 규칙**
  - 소멸자 이름은 클래스 이름과 같아야 하며 ~를 함께 달아야 한다
  - 소멸자는 인수와 반환 값이 없다. (생성자와 달리 매개변수를 가질 수 없다)
- 이러한 규칙 때문에 소멸자는 **클래스당 하나** 밖에 존재할 수 없다.
- 소멸자를 명시적으로 호출하는 경우는 코드 재사용할 때 외에는 없다.
  - 대상 클래스가 무엇을 담고 있는지 꼭 확인 후 사용해야 문제가 없음!

소멸자 예제 [2/2]

- Person 클래스: 소멸자를 사용한 예제

```
1.  class Person {
2.      string name;
3.      int age;
4.  public:
5.      Person() { name = "동국"; age = 20; }
6.      Person(string n) { name = n; age = 24; }
7.      Person(string n, int a) { name = n; age = a; }
8.      string getName() { return name; }
9.      int getAge() { return age; }
10.     ~Person() { cout << name << "님의 기록이 삭제되었습니다." << endl; }
11. };
12.
13. int main() {
14.     Person ps;
15.     Person pt("유리");
16.     Person pa("철수", 22);
17.     cout << "이름 : " << ps.getName() << ", 나이 : " << ps.getAge() << endl;
18.     cout << "이름 : " << pa.getName() << ", 나이 : " << pa.getAge() << endl;
19.     cout << "이름 : " << pt.getName() << ", 나이 : " << pt.getAge() << endl;
20. }
```

연습문제(3장)

1.

객체를 캡슐화하는 목적은 무엇인가?

2.

클래스와 객체에 관한 설명 중 틀린 것은?

① 객체를 실체 혹은 인스턴스(instance)라고 부른다.
② 클래스는 객체를 생성하기 위한 설계도 혹은 틀과 같다.
③ 클래스의 멤버들은 private보다 public 접근 지정이 바람직하다.
④ 클래스는 함수 멤버와 변수 멤버로 이루어진다.

3.

다음 C++ 코드가 객체 지향 언어의 캡슐화를 달성하고 있는지 설명하라.

```
int acc;
int add(int x) {
   acc += x;
   return acc;
}
class Circle {
public:
   int radius;
   double getArea();
};
```

연습문제(3장)

4. 다음 C++ 프로그램에 캡슐화가 부족한 부분을 수정하여 캡슐화하라.

```
int age;
void older() {
   age++;
}
class Circle {
   int radius;
public:
   double getArea();
};
```

5. 다음 코드는 Circle 클래스의 선언부이다. 틀린 부분을 수정하라.

```
class Circle {
   int radius;
   double getArea();
}
```

연습문제(3장)

6. 다음 코드는 Tower 클래스를 작성한 사례이다. 틀린 부분을 수정하라.

```
class Tower {
   int height = 20;
public:
   Tower() { height = 10; return; }
};
```

7. 다음 코드에서 틀린 부분을 수정하라.

```
class Building {
private:
   int floor;
public:
   Building(int s) { floor = s; }
};
int main() {
   Building twin, star;
   Building BlueHouse(5), JangMi(14);
}
```

연습문제(3장)

8. 다음 코드는 Calendar 클래스의 선언부이다. year를 10으로 초기화하는 생성자와 year 값을 리턴하는 getYear()를 구현하라.

```
class Calendar {
private:
   int year;
public:
   Calendar();
   int getYear();
};
```

9. 생성자에 대한 설명 중 틀린 것은?

① 생성자의 이름은 클래스 이름과 같다.
② 생성자는 오직 하나만 작성 가능하다.
③ 생성자는 리턴 타입을 가지지 않는다.
④ 생성자가 선언되어 있지 않으면 컴파일러에 의해 기본 생성자가 삽입된다.

10. 소멸자에 대한 설명 중에 틀린 부분을 지적하라.

소멸자는 ① 객체가 소멸되는 시점에 자동으로 호출되는 멤버 함수로서 ② 클래스의 이름 앞에 ~를 붙인 이름으로 선언되어야 한다. ③ 매개 변수 있는 소멸자를 작성하여 소멸 시에 의미 있는 값을 전달할 수 있으며, 소멸자가 선언되어 있지 않으면 ④ 기본 소멸자가 자동으로 생성된다.

연습문제(3장)

11. 다음 프로그램에 대해 답하여라.

```
class House {
   int numOfRooms;
   int size;
public:
   House(int n, int s); // n과 s로 numOfRooms, size를 각각 초기화
};
void f() {
   House a(2,20);
}
House b(3,30), c(4,40);
int main() {
   f();
   House d(5,50);
}
```

(1) n과 s로 numOfRooms, size를 각각 초기화하고, 이들을 출력하는 생성자를 구현하라.

(2) size와 numOfRooms 값을 출력하는 House 클래스의 소멸자를 작성하라.

(3) 객체 a,b,c,d가 생성되는 순서와 소멸되는 순서는 무엇인가?

연습문제(3장)

12. 다음 프로그램에서 객체 a,b,c가 생성되고 소멸되는 순서는 무엇인가?

```
class House {
   int numOfRooms;
   int size;
public:
   House(int n, int s) { numOfRooms = n; size = s; }
   void test() {
      House a(1,10);
   }
};
void f() {
   House b(2,20);
   b.test();
}
House c(3,30);
int main() {
   f();
}
```

연습문제(3장)

13. 다음 프로그램의 오류를 지적하고 수정하라.

```
class TV {
   TV() { channels = 256; }
public:
   int channels;
   TV(int a) { channels = a;}
};
int main() {
   TV LG;
   LG.channels = 200;
   TV Samsung(100);
}
```

연습문제(3장)

14. 다음 프로그램의 오류를 지적하고 수정하라.

```
class TV {
   int channels;
public:
   int colors;
   TV() { channels = 256; }
   TV(int a, int b) { channels = a; colors = b; }
};
int main() {
   TV LG;
   LG.channels = 200;
   LG.colors = 60000;
   TV Samsung(100, 50000);
}
```

연습문제(3장)

15. 다음 코드에서 자동 인라인 함수를 찾아라.

```
class TV {
   int channels;
public:
   TV() { channels = 256; }
   TV(int a) { channels = a; }
   int getChannels();
};
inline int TV::getChannels() { return channels; }
```

16. 인라인 함수의 장단점을 설명한 것 중 옳은 것은?

① 인라인 함수를 사용하면 컴파일 속도가 향상된다.
② 인라인 함수를 이용하면 프로그램의 실행 속도가 향상된다.
③ 인라인 함수를 사용하면 프로그램 작성 시간이 향상된다.
④ 인라인 함수를 사용하면 프로그램의 크기가 작아져서 효과적이다.

연습문제(3장)

17. 인라인 함수에 대해 잘못 설명한 것은?

① 인라인 선언은 크기가 큰 함수의 경우 효과적이다.

② C++ 프로그램에는 크기가 작은 멤버 함수가 많기 때문에 이들을 인라인으로 선언하면 효과적이다.

③ 컴파일러는 먼저 인라인 함수를 호출하는 곳에 코드를 확장시킨 후 컴파일한다.

④ 인라인 함수는 함수 호출에 따른 오버헤드를 줄이기 위한 방법이다.

18. inline 선언은 강제 사항이 아니다. 다음 함수 중에서 컴파일러가 인라인으로 처리하기에 가장 바람직한 것은?

①
```
inline int big(int a, int b) {
   return a>b?a:b;
}
```

②
```
inline int sum(int a, int b) {
   if(a>=b) return a;
   else return a + sum(a+1, b);
}
```

③
```
inline void add(int a, int b) {
   int sum=0;
   for(int n=a; n<b; n++)
      sum += n;
}
```

④
```
inline int add(int a) {
   static int x = 0;
   x += a;
   return x;
}
```

연습문제(3장)

19. C++ 구조체(struct)에 대해 잘못 설명한 것은?

① C++에서 구조체를 둔 이유는 C 언어와의 호환성 때문이다.
② C++에서 구조체는 멤버 함수와 멤버 변수를 둘 수 있다.
③ C++에서 구조체는 생성자와 소멸자를 가진다.
④ C++에서 구조체는 상속을 지원하지 않는다.

20. 다음 C++ 구조체를 동일한 의미를 가지는 클래스로 작성하라.

```
struct Family {
   int count;
   char address[20];
public:
   Family();
private:
   char tel[11];
};
```

21. 다음 클래스를 구조체로 선언하라.

```
class Universe {
   char creator[10];
   int size;
private:
   char dateCreated[10];
public:
   Universe();
};
```

실습과제(3장)

1. main()의 실행 결과가 다음과 같도록 Tower클래스를 작성하라.

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

int main() {
    Tower myTower; // 1 미터
    Tower seoulTower(100); // 100 미터
    cout << "높이는" << myTower.getHeight() << "미터" << endl;
    cout << "높이는" << seoulTower.getHeight() << "미터" << endl;
}
```

```
높이는 1미터
높이는 100미터
```

실습과제(3장)

2. 날짜를 다루는 Date 클래스를 작성하고자 한다. Date를 이용하는 main()과 실행결과는 다음과 같다. 클래스 Date를 작성하여 아래 프로그램에 추가하라.

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

int main() {
    Date birth(2014, 3, 20); // 2014년 3월 20일
    Date independenceDay("1945/8/15"); // 1945년 8월 15일
    independenceDay.show();
    cout << birth.getYear() << "," << birth.getaMonth() << "," << birth.getDay() << endl;
}
```

```
1945년8월15일
2014, 3, 20
```

<string>헤더 파일의 stoi() 함수를 이용하면 string의 문자열을 숫자로 변환할 수 있다. stoi()는 C++11 표준부터 삽입되었다.

```
string s = "1945";
int n = stoi(s); // n은 정수 1945. VS 2008에는 int n = atoi(s.c_str());
```

실습과제(3장)

3. 은행에서 사용하는 프로그램을 작성하기 위해, 은행 계좌 하나를 표현하는 클래스 Account가 필요하다. 계좌 정보는 계좌의 주인, 계좌 번호, 잔액을 나타내는 3 멤버 변수로 이루어진다. main() 함수의 실행 결과가 다음과 같도록 Account 클래스를 작성하라.

```
int main() {
    Account a("kitae", 1, 5000); // id 1번, 잔액 5000원, 이름이 kitae인 계좌 생성
    a.deposit(50000); // 50000원 저금
    cout << a.getOwner() << "의 잔액은" << a.inquiry() << endl;
    int money = a.withdraw(20000); // 20000원 출금. withdraw()는 출금한 실제 금액 리턴
    cout << a.getOwner() << "의 잔액은" << a.inquiry() << endl;
}
```

```
kitae의 잔액은 55000
kitae의 잔액은 35000
```

Account는 name, id, balance(잔액)의 3 멤버 변수와 생성자, getOwner(), deposit(), withdraw(), inquiry()의 3 멤버 함수를 가지는 클래스로 만들면 된다.

실습과제(3장)

4. CoffeeMachine 클래스를 만들어보자. main() 함수와 실행 결과가 다음과 같도록 CoffeeMachine 클래스를 작성하라. 에스프레소 한 잔에는 커피와 물이 각각 1씩 소비되고, 아메리카노의 경우 커피는 1, 물은 2가 소비되고, 설탕 커피는 커피 1, 물 2, 설탕 1이 소비된다. CoffeeMachine 클래스에는 어떤 멤버 변수와 어떤 멤버 함수가 필요한지 잘 판단하여 작성하라.

```
int main () {
    CoffeeMachine java(5, 10, 3): // 커피량:5, 물량:10, 설탕:6으로 초기화
    java.drinkespresso(); // 커피 1, 물 1 소비
    java.show(); // 현재 커피 머신의 상태 출력
    java.dninkAmericano(): // 커피 1, 물 2 소비
    java.show(): // 현재 커피 머신의 상태 출력
    java.drinksugarCoffee(); // 커피 1, 물 2, 설탕 1 소비
    java.show(); // 현재 커피 머신의 상태 출력
    java.fill(); // 커피 10, 물 10, 설탕 10으로 채우기
    java.show(); // 현재 커피 머신의 상태 출력
}
```

```
커피 머신 상태, 커피:4 물:9          설탕: 3
커피 머신 상태, 커피:3 물:7          설탕: 3
커피 머신 상태, 커피:2 물:5          설탕: 2
커피 머신 상태, 커피: 10 물: 10      설탕: 10
```

실습과제(3장)

5. 랜덤 수를 발생시키는 Random 클래스를 만들자. Random 클래스를 이용하여 랜덤한 정수를 10개 출력하는 사례는 다음과 같다. Random 클래스가 생성자, next(), nextInRange()의 3개의 멤버 함수를 가지도록 작성하고 main() 함수와 합쳐 하나의 cpp 파일에 구현하라.

```
int main () {
    Random r;
    cout <<"-- 0에서 "<< RAND_MAX << "까지의 랜덤 정수 10 개--" << endl;
    for (int i=0; i<10; i++) {
        int n = r.next(): // 0에서 RAND_MAX(32767) 사이의 랜덤한 정수
        cout << n << ' ';
    }
    cout << endl << endl  << "-- 2에서 " <<"4 까지의 랜덤 정수 10--" << endl ;
    for (int i=0; i<10; i++) {
        int n = r.nextInRange(2, 4): // 2에서 4 사이의 랜덤한 정수
        cout << n << ' ' ;
    }
    cout << endl;
}
```

```
-- 0에서 32767까지의 랜덤 정수 10 개--
3975 1512 15096 4317 14047 30968 21702 24510 5530 6633
-- 2에서 4 까지의 랜덤 정수 10 개--
2 3 3 2 2 4 2 3 4 2
```

랜덤 정수를 발생시키기 위해 다음 두 라인의 코드가 필요하고, <cstdlib>와 <ctime> 헤더 파일을 include해야 한다.

```
srand ((unsigned)time(0)); // 시작할 때마다, 다른 랜덤수를 발생시키기 위한 seed 설정
int n= rand(); // 0에서 RAND_MAX (32767) 사이의 랜덤한 정수 발생
```

실습과제(3장)

6. 문제 5번을 참고하여 짝수 정수만 랜덤하게 발생시키는 EvenRandom 클래스를 작성하고 EvenRandom 클래스를 이용하여 10개의 짝수를 랜덤하게 출력하는 프로그램을 완 성하라. 0도 짝수로 처리한다.

```
-- 0에서 32767까지의 랜덤 정수 10 개--
4556 9038 18662 27744 14522 2832 8594 18428 4130 24854
-- 2에서 10 까지의 랜덤 정수 10 개--
8 4 8 6 8 10 6 10 6 8
```

7. 문제 5번을 참고하여 생성자를 이용하여 짝수 홀수를 선택할 수 있도록 Selectable Random 클래스를 작성하고 짝수 10개, 홀수 10개를 랜덤하게 발생시키는 프로그램을 작성하라.

```
-- 0에서 32767까지의 짝수 랜덤 정수 10 개 --
15626 13266 28780 17984 16836 218 2020 24888 8532 9748
-- 2에서 9까지의 랜덤 홀수 정수 10 개--
7 9 7 5 5 9 3 3 9 7
```

실습과제(3장)

8. int 타입의 정수를 객체화한 Integer 클래스를 작성하라. Integer의 모든 멤버 함수를 자동 인라인으로 작성하라. Integer 클래스를 활용하는 코드는 다음과 같다.

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

int main() {
Integer n(30);
cout << n.get() << ' '; // 30 출력
n. set (50);
cout << n.get() << ' '; // 50 출력

Integer m("300");
cout << m.get() << ' '; // 300 출력
cout << m.isEven(); // true(정수로 1) 출력
 }
```

```
30 50 300 1
```

문제 2의 힌트와 동일하게 <string> 헤더 파일의 stoi() 함수를 이용하면 편하다.

실습과제(3장)

9. Oval 클래스는 주어진 사각형에 내접하는 타원을 추상화한 클래스이다. Oval 클래스의 멤버는 모두 다음과 같다. Oval 클래스를 선언부와 구현부로 나누어 작성하라.
   - 정수값의 사각형 너비와 높이를 가지는 width, height 변수 멤버
   - 너비와 높이 값을 매개 변수로 받는 생성자
   - 너비와 높이를 1로 초기화하는 매개 변수 없는 생성자
   - width와 height를 출력하는 소멸자
   - 타원의 너비를 리턴하는 getwidth() 함수 멤버
   - 타원의 높이를 리턴하는 getHeight() 함수 멤버
   - 타원의 너비와 높이를 변경하는 set (int w, int h) 함수 멤버
   - 타원의 너비와 높이를 화면에 출력하는 show() 함수 멤버

- Oval 클래스를 활용하는 코드의 사례와 실행 결과는 다음과 같다.

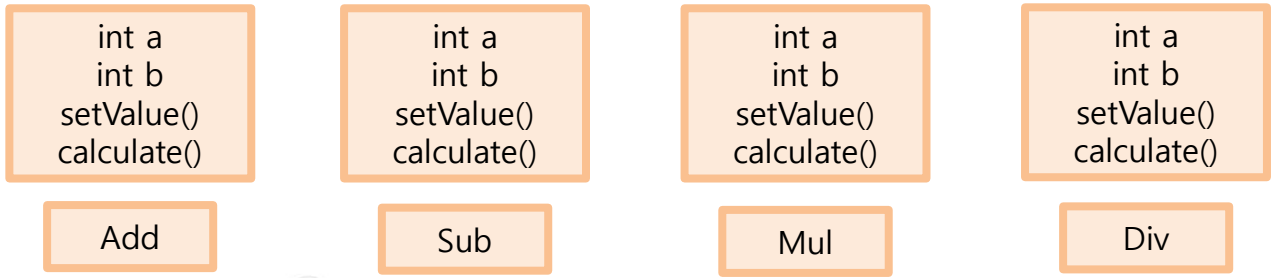
```
#include <iostream>
using namespace std;

int main () {
Oval a, b(3, 4);
a.set (10, 20);
a.show();
cout << b.getWidth() <"," <<  b.getHeight() << endl;
```

a.show(); 의 실행 결과

```
width = 10, height = 20
3, 4
Oval 소멸 : width = 3, height = 4
Oval 소멸 : width = 10, height = 20
```

실습과제(3장)

10. 다수의 클래스를 선언하고 활용하는 간단한 문제이다. 더하기(+), 빼기(-), 곱하기 (*), 나누기(/)를 수행하는 4개의 클래스 Add, Sub, Mul, Div를 만들고자 한다. 이들 은 모두 공통으로 다음 멤버를 가진다.
    - int 타입 변수 a, b : 피연산자
    - void setValue(int x, int y) 함수 : 매개 변수 x, y를 멤버 a, b에 복사
    - int calculate() 함수 : 연산을 실행하고 결과 리턴



main() 함수는 Add, Sub, Mul, DiV 클래스 타입의 객체 a, s, m, d를 생성하고, 아래 와 같이 키보드로부터 두 개의 정수와 연산자를 입력받고, a, s, m, d 객체 중에서 연 산을 처리할 객체의 setValue() 함수를 호출한 후, calculate()를 호출하여 결과를 화면에 출력한다. 프로그램은 무한 루프를 돈다.

실습과제(3장)

10.

```
두 정수와 연산자를 입력하세요 >> 3 4 *
12
두 정수와 연산자를 입력하세요>>5 2 /
2
두 정수와 연산자를 입력하세요>>8 2 -
6
```

(1) 클래스의 선언부와 구현부를 분리하고, 모든 코드를 Calculator.cpp 파일에 작성하라.

(2) 클래스의 선언부와 구현부를 헤더 파일과 cpp 파일로 나누어 프로그램을 작성하라.

실습과제(3장)

11. 다음 코드에서 Box 클래스의 선언부와 구현부를 Box.h, Box.cpp 파일로 분리하고 main() 함수 부분을 main.cpp로 분리하여 전체 프로그램을 완성하라.

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

class Box {
int width, height;
char fill; // 박스의 내부를 채우는 문자
public:
Box (int w, int h) { setSize(w, h); fill = '*'; }
void setFill(char f) { fill = f; }
void setsize(int W, Int h) { width =w; height = h; }
void draw(): // 박스 그리기
 }  ;
void Box:: draw() {
for(int n=0; n<height; n+) 1
for (int m=0; m<width; m++) cout << fill;
cout << endl;
 }
int main () {
Box b (10, 2);
b.draw();  // 박스를 그린다.
cout << endl;
b.setSize(7, 4); // 박스의 크기를 변경한다.
b.setFill('^');  // 박스의 내부를 채울 문자를 '^'로 변경한다.
b.draw(); // 박스를 그린다.
}
```

실습과제(3장)

```
**********
**********

^^^^^^^
^^^^^^^
^^^^^^^
^^^^^^^
```

실습과제(3장)

12. 컴퓨터의 주기억장치를 모델링하는 클래스 Ram을 구현하려고 한다. Ram 클래스는 데이터가 기록될 메모리 공간과 크기 정보를 가지고, 주어진 주소에 데이터를 기록하고 (write), 주어진 주소로부터 데이터를 읽어 온다 (read), Ram 클래스는 다음과 같이 선언된다.

```
class Ram {
char mem[100*1024]; // 100KB 메모리. 한 번지는 한 바이트이므로 char 타입 사용
int size;
public:
Ram(); // mem 배열을 0으로 초기화하고 5ize 100*1024로 초기화
~Ram(); // "메모리 제거됨" 문자열 출력
char read(int address); // address 주소의 메모리 바이트 리턴
void write(int address, char value): // address 주소에 한 바이트로 value 저장
};
```

실습과제(3장)

12. 다음 main() 함수는 100 번지에 20을 저장하고, 101 번지에 30을 저장한 후, 100 번지와 101 번지의 값을 읽고 더하여 102 번지에 저장하는 코드이다.

```
int main () {
Ram ram;
ram write(100, 20); // 100 번지에 20 저장 ram.write(101, 30); // 101 번지에 30 저장
char res = ram. read 100) + ram. read (101); // 20 + 30 = 50
ram.write(102, res); // 102 번지에 50 저장
cout << "102 번지의 값 =" << (int)ram.nead (102) < endl; // 102 번지 값 출력
```

```
102 번지의 값 = 50
메모리 제거됨
```

실행 결과를 참고하여 Ram.h, Ram. cpp, main.cpp로 헤더 파일과 cpp 파일을 분리하여 프로그램을 완성하라.