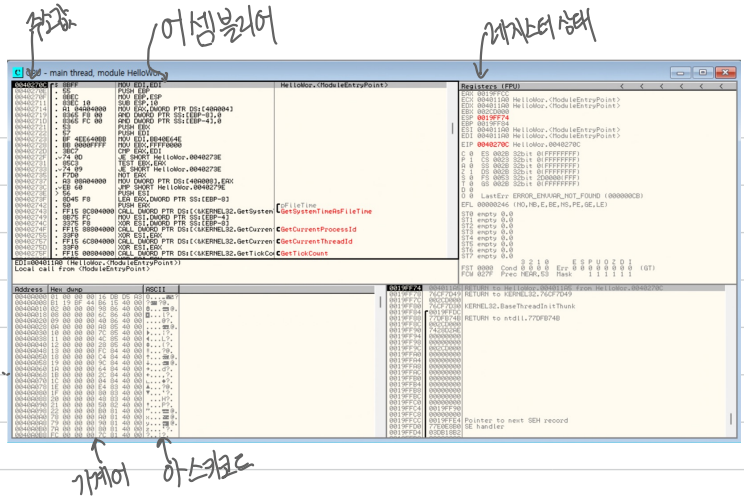
2주차 C++ 수업 내용 정리



1. 프로그래밍 언어 토의

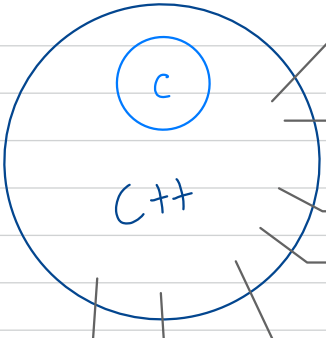
- **기계어** (이진수) 기계는 기계어만 사용가능
- **어셈블러어** 기계어를 니모닉 언어와 일대일 대응
  - ↳ **어셈블러** 어셈블러어 → 기계어 (변환)
- **고급언어** 사람이 이해하기 쉽게 만든언어 ex) Java, C/C++
  - ↳ **컴파일러** 고급언어 → 기계어

2. C++ 표준화

- 1998 미국 표준원 ISO/IEC 문서작성
- 진화 순서 C++98(1998)
  - ↓
  - C++03(2003)
  - ↓
  - C++TR1(2007)
  - ↓
  - C++11(2011)

표준의 장점
- 모든 컴파일러에서 동일한 값 도출
  ∋ 호환성↑

비표준 현황
- Visual C++/Borland C++ ⇒ 특정 컴파일러에서만 작동
  ∴ 호환성 ↓



함수 중복 매개변수의 종류나 개수, 타입이 다르면 함수의 이름이 같아도 됨

디폴트 매개변수 매개변수에 디폴트값이 전달되도록 함수선언 Ok!

참조와 참조변수 하나의 변수에 별명을 사용

Call-by-Reference (C언어에도 포인터를 사용해 이 기능사용가능)

new/delete 연산자 동적 메모리 해제/할당 ∋ C언어에서는 malloc과 free이용

연산자 재정의 기존 연산자에 새로운 연산 정의

제네릭 함수와 클래스 데이터 타입에 의존하지 않고 일반화시킨 함수나 클래스 작성 ok!

2주차 C++ 수업 내용정리(2)

3. C++ 객체지향 특성

1) 캡슐화

- 데이터를 캡슐화하여 외부 접근으로부터 보호 (class)

클래스 - 객체를 만드는 틀

객체 - 클래스라는 틀에서 만들어진 실체

2) 상속성

객체가 자식 클래스의 멤버와 부모 클래스에 선언된 모양 그대로 멤버들을 가지고 탄생

3) 다형성

하나의 기능이 경우에 따라 다르게 보이거나 다르게 작동하는 현상 ex) 연산자 중복, 함수 중복, 함수 재정의

도입 목적

- 소프트웨어 생산성 향상 - 소프트웨어 생명주기 단축문제 해결 필요
  - 코드의 재사용 필요
- 실세계 필요에 맞게 변화
  - 과거: 절차지향 프로그래밍 (수학/통계계산)
  - 현재: 객체지향 프로그래밍 (실세계는 객체로 구성)

3주차 C++ 수업내용 정리

주석문 : 프로그램 실행에 영향 X

<iostream>헤더 파일 : 표준 입출력을 위한 클래스와 객체, 변수 등이 선언됨

main()함수: int main(){

↳ ios, istream, ostream, iostream클래스 선언

↳ cout, cin, <<, >> 등 연산자

return 0; ⟸ return 문은 생략가능

}

cout 객체 : 스크린 출력 장치에 연결된 표준 C++ 출력 스트림 객체

↳ std::cout 으로 사용

<< : 스트림 삽입 연산자 → ostream 클래스

C++ 기본 산술 시프트 연산자 ⇒ 스트림 삽입 연산자로 재정의

문자열 및 기본 타입의 데이터 출력 : bool, char, short, int, long, float, double 타입 출력 / 함수호출도 가능 / '\n' or std::endl 사용

namespace 개념

↳ 이름 충돌 문제 해결! (개발자가 자신만의 이름 공간을 생성할 수 있도록함)

1) std : <iostream> 헤더파일에 선언된 모든 이름 → std 이름 공간에 있음 참고를 위해 std:: 접두어 사용

using 지시어 사용 ① using std::cout ② using namespace std

2) cin : 표준 입력 장치인 키보드를 연결하는 C++ 입력 스트림 객체

>> (스트림 추출 연산자) : 입력 스트림에서 값을 읽어 변수에 저장

동작 입력 버퍼 내용 /<Enter>키가 입력될 때까지 입력된 키를 입력버퍼에 저장

cin.getline() : 공백이 낀 문자열을 입력 받는 방법

C++ 변수 선언

· 어무곳에나 가능 장점 변수 사용 장전 선언→더이밍문제! 단점 선언된 변수 일괄적으로 보기 힘듬/코드 사이 변수 찾기 어려움

C++ 문자열 표현방식

1. C-스트링 방식 - '\0'로 끝나는 문자배열 <string.h> or <cstring> 헤더 사용

2. string 클래스 이용 - <string> 헤더 파일에 선언 / 다양한 멤버함수 제공, 문자열 비교, 복사, 수정 등

String 클래스

- C#에서 참조 형식 / C# 표준 클래스 / 문자열의 크기에 따른 제약 없음(스스로 문자열 크기에 맞게 메모리 할당) / 객체지향적

클래스와 객체

·캡슐화 → 객체를 캡슐로 싸서 그 내부를 보호하고 볼 수 없게함 = (중요) 객체 내 데이터에 대한 보안, 보호, 외부접근제한

·객체의 일부분 공개 : 외부와의 인터페이스를 위해 객체 일부분 공개

C++ 객체 → 멤버함수 & 멤버 변수 로 구성

→ 생성될때 클래스 모양 그대로 가지고 탄생

→ 메모리에 생성 (instance)

→ 하나의 클래스 ⇒ 여러 객체 생성 / 객체들은 서로 별도의 영역에 멤버를 만든다.

C++클래스 → 객체를 만들어내기 위해 정의된 설계도

→ 멤버 변수 & 멤버 함수 선언

선언부 → class 키워드사용 / 멤버함수는 원형형태로 선언 / 디폴트 private, public, protected 중 하나

구현부 → 클래스에 정의된 모든 멤버함수 구현

예제 3-1 Circle 클래스의 객체 생성 및 활용

Ex3_1.cpp

```cpp
#include <iostream>
using namespace std;
class Circle
{
public:
    int radius;
    double getArea();
};
double Circle::getArea()
{
    return 3.14 * radius * radius;
}
int main()
{
    Circle donut;
    donut.radius = 1;                   // donut 객체의 반지름을 1로 설정
    double area = donut.getArea(); // donut 객체의 면적 알아내기
    cout << "donut 면적은 " << area << endl;

    Circle pizza;
    pizza.radius = 30;      // pizza 객체의 반지름을 30으로 설정
    area = pizza.getArea(); // pizza 객체의 면적 알아내기
    cout << "pizza 면적은 " << area << endl;

    return 0;
}
```

Circle 선언부

Circle 구현부

객체

```
문제  출력  디버그 콘솔  터미널  포트

cd "/Users/leejoohyun/Desktop/2학기/객체지향프로그래밍/Week4/" && g++ Ex3_1.cpp -o Ex3_1 && "/Users/leejoohyun/Desktop/2학기/객체지향프로그래밍/Week4/"Ex3_1
leejoohyun@ijuhyeon-ui-MacBookAir Week4 % cd "/Users/leejoohyun/Desktop/2학기/객체지향프로그래밍/Week4/" && g++ Ex3_1.cpp -o Ex3_1 && "/Users/leejoohyun/Desktop
/2학기/객체지향프로그래밍/Week4/"Ex3_1
donut 면적은 3.14
pizza 면적은 2826
leejoohyun@ijuhyeon-ui-MacBookAir Week4 %
```

객체의 멤버함수 & 멤버변수 접근 ⇒ '.' 연산자 사용

예제 3-2(실습)-Rectangle 클래스 만들기


```
Ex3_2.cpp > main()
#include <iostream>
using namespace std;
class Rectangle
{
public:
    int width, height, getArea();
};
int Rectangle::getArea()
{
    return width * height;
}
int main()
{
    Rectangle rect;
    rect.width = 3;
    rect.height = 5;
    cout << "사각형의 면적은 " << rect.getArea() << endl;
    return 0;
}
leejoohyun@jubyeon-ui-MacBookAir Week4 % cd "/Users/leejoohyun/Desktop/2학기/객체지향프로그래밍/Week4/" && g++ Ex3_2.cpp -o Ex3_2 && "/Users/leejoohyun/Desktop/2학기/객체지향프로그래밍/Week4/"Ex3_2
사각형의 면적은 15
leejoohyun@jubyeon-ui-MacBookAir Week4 %
```

클래스 구현시 주의할 점은 디폴트값은 private이기 때문에 public으로 선언을 해주지 않는다면 원하는 결과 X

생성자 : 객체가 생성되는 시점에서 자동으로 호출되는 멤버함수(클래스 이름과 동일)

- 목적 : 객체가 생성될 때 객체가 필요한 초기화를 위해
- 리턴 타입 선언 X / 객체 생성시 한 번만 호출
- 중복 가능 / 생성자 선언되어 있지 X → 기본 생성자 자동생성 || 선언되어 있으면 기본생성자 자동 생성 X

예제 3-3 2개의 생성자를 가진 Circle 클래스

```
Ex3_3.cpp > Circle(int)
#include <iostream>
using namespace std;
class Circle
{
public:
    int radius;
    Circle();       // 매개 변수 없는 생성자
    Circle(int r); // 매개 변수 있는 생성자
    double getArea();
};
Circle::Circle()
{
    radius = 1;
    cout << "반지름 " << radius << " 원 생성" << endl;
}
Circle::Circle(int r)
{
    radius = r;
    // this->radius = r;
    cout << "반지름 " << radius << " 원 생성" << endl;
}
double Circle::getArea()
{
    return 3.14 * radius * radius;
}
```

타겟 생성자 : 객체 초기화를 전담하는 생성자

위임 생성자 : 타겟 생성자를 호출하는 생성자, 객체 초기화를 타겟 생성자에 위임

예제 3-4 생성자에서 다른 생성자 호출 연습

```
Ex3_4.cpp > main()
#include <iostream>
using namespace std;
class Circle
{
public:
    int radius;
    Circle();       // 위임 생성자
    Circle(int r); // 타겟 생성자
    double getArea();
};
Circle::Circle() : Circle(1) {} // 위임 생성자
Circle::Circle(int r){
    radius = r;
    cout << "반지름 " << radius << " 인 원 생성" << endl;
}
double Circle::getArea(){
    return 3.14 * radius * radius;
}
int main(){
    Circle donut; // 매개 변수 없는 생성자 호출
    double area = donut.getArea();
    cout << "donut 면적은 " << area << endl;
    Circle pizza; // 매개 변수 있는 생성자 호출
    area = pizza.getArea();
    cout << "pizza 면적은 " << area << endl;
    return 0;
}
```

```
반지름 1 인 원 생성
donut 면적은 3.14
반지름 1 인 원 생성
pizza 면적은 3.14
```

다양한 생성자의 멤버변수 초기화 방법

(1) 생성자 코드에서 멤버 변수 초기화 Point::Point(){x=0;y=0;}

Point::Point(int a, int b){x=a;y=b;}

(2) 생성자 서두에 초깃값으로 초기화 Point::Point():x(a),y(b){}

Point::Point(int a, int b):x(a),y(b){}

(3) 클래스 선언부에서 직접 초기화 class Point{

public:
int x=0,y=0;
};

예제 3-5 멤버변수의 초기화와 위임 생성자 활용

Ex3_5.cpp > Point(int, int)

```cpp
#include <iostream>
using namespace std;
class Point{
    int x, y;
public:
    Point();
    Point(int a, int b);
    void show() { cout << "(" << x << ", " << y << ")" << endl; }
};
Point::Point(){
    x = 0;
    y = 0;
} // 위임 생성자
Point::Point(int a, int b){
    x = a;
    y = b;
} // 타겟 생성자
/*
Point::Point():Point(0,0){}
Point::Point(int a, int b):x(a),y(b){}
*/
int main(){
    Point origin;
    Point target(10, 20);
    origin.show();
    target.show();
}
```

```
leejoohyun@ijuhyeon-ui-MacBookAir Week4 %
/2학기/객체지향프로그래밍/Week4/"Ex3_5
(0, 0)
(10, 20)
leejoohyun@ijuhyeon-ui-MacBookAir Week4 %
```

·기본 생성자 -클래스에 생성자가 하나도 선언되어 있지 않은 경우, 컴파일러가 대신 삽입해주는 생성자

- 매개 변수 없는 생성자 (디폴트 생성자)

예제 3-6(실습) - Rectangle 클래스 만들기

Ex3_6.cpp > ...

```cpp
#include <iostream>
using namespace std;
class Rectangle{
public:
    int x, y;
    Rectangle();
    Rectangle(int c);
    Rectangle(int a, int b);
    bool isSquare();
};
Rectangle::Rectangle(){
    x = 0;
    y = 0;
}
Rectangle::Rectangle(int a, int b){
    x = a;
    y = b;
}
Rectangle::Rectangle(int c){
    x = y = c;
}

bool Rectangle::isSquare(){
    return x == y ? true : false;
}
int main()
{
    Rectangle rect1;
    Rectangle rect2(3, 5);
    Rectangle rect3(3);

    if (rect1.isSquare())
        cout << "rect1은 정사각형이다." << endl;
    if (rect2.isSquare())
        cout << "rect2은 정사각형이다." << endl;
    if (rect3.isSquare())
        cout << "rect3은 정사각형이다." << endl;
    return 0;
}
```

```
leejoohyun@ijuhyeon-ui-MacBookAir Week4 %
/2학기/객체지향프로그래밍/Week4/"Ex3_6
rect1은 정사각형이다.
rect3은 정사각형이다.
leejoohyun@ijuhyeon-ui-MacBookAir Week4 %
```

소멸자 : 객체가 소멸되는 시점에서 자동으로 호출되는 함수 (객체 메모리 소멸 직전 호출됨)

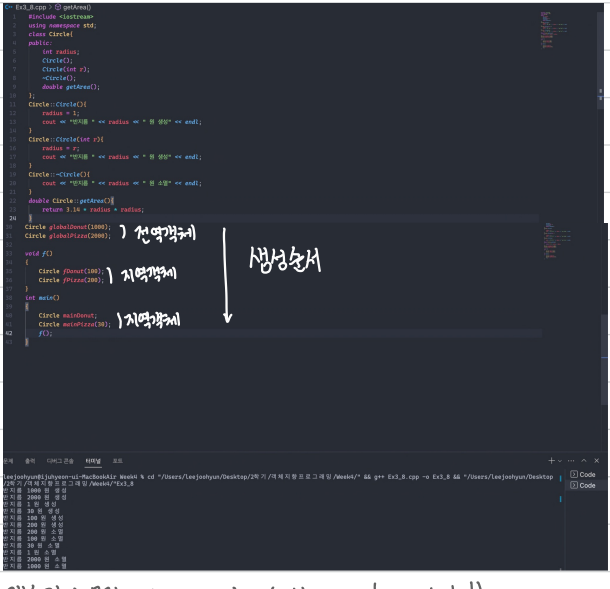
· 목적 : 할당받은 메모리 해제, 파일 저장 및 닫기 등

· 특징 : 리턴 타입 선언 불가 / 소멸자가 선언되어있지 X → 자동 생성

객체 생성순서 → 선언된 순서 / 객체 소멸순서 → 선언된 순서의 역순

new (객체 생성) ↓ delete (객체 소멸)

예제 3-8 지역객체와 전역 객체의 생성 및 소멸 순서



캡슐화의 목적 : 객체 보호, 보안 (public, private, protected)

인라인 함수 : inline 키워드로 선언된 함수 (컴파일러에 의해 이루어짐)

└ 목적 : C++ 프로그램의 실행속도↑↑ (장점)

단점 : 컴파일되는 코드 전체 크기 증가

C++ 구조체 ≒ Class

↳ 사용이유 : C언어의 호환성을 위해