객체 지향 프로그래밍
(Object Oriented Programming)

12장

강사 – 강대기

차례 (Agenda)

- 멤버에 대한 동적 메모리 할당
- 암시적/명시적 복사 생성자
- 암시적/명시적 오버로딩 대입 연산자
- 생성자에 new 사용하기
- static 클래스 멤버
- 객체에 위치지정 new 사용하기
- 객체를 지시하는 포인터

StringBad 클래스

- 멤버에 포인터 사용 – str
- static 멤버 num_strings – 단 하나만 생성됨
- static 클래스 멤버 초기화 방법 (713쪽)
  - int StringBad::num_strings = 0;
  - const나 enum을 사용하여 클래스 선언 안에서 초기화 (619쪽)
- 생성자에서 new[]를 사용하면 파괴자에서 delete[]를 사용

StringBad 클래스 사용 시의 문제점 – vegnews.cpp (717쪽)

- callme2(headline2);
  - 값으로 함수 인자 전달 – 임시객체 생성, 얕은 복사 수행
- StringBad sailor = sports;
  - 새로 정의하는 객체에 대입
  - 복사 생성자를 사용하거나, 복사 생성자로 임시 객체 생성 후 대입 연산자 사용
  - 복사 생성자, 대입 연산자, 얕은 복사 수행
- knot = headline1;
  - 다른 객체에 대입 – 대입 연산자 사용, 얕은 복사 수행

컴파일러가 몰래 만드는 멤버 함수

- 생성자가 없을 경우, 디폴트 생성자
- 복사 생성자
- 대입 연산자
- 파괴자

- 주소 연산자 – operator&
- 콤마 연산자 – operator,

디폴트 생성자

- 어떠한 생성자도 만들지 않으면 생기며, 전달 인자는 없다
- 다른 생성자를 만들면, 없어짐
- 명시적으로 디폴트 생성자를 만들거나
- 디폴트 전달 인자를 사용하여 구현
- 개선 방안
  - this->str = new char[1]; 로 초기화함

복사 생성자 (723쪽)

- 어떤 객체를 받아서 그 내용을 그대로 복사해서, 새로운 객체를 생성
- Class_name(const Class_name&);
- StringBad(const StringBad&);
- 호출되는 예
  - StringBad ditto(motto);
  - StringBad metoo = motto;
  - StringBad also = StringBad(motto);
  - StringBad* pS = new StringBad(motto);
- 또한 객체를 함수의 전달 인자로 값으로 전달할 때
- 복사 생성자도 생성자이므로 new/초기화 수행

대입 연산자 (operator=) (728쪽)

- 객체에 다른 객체를 대입
- Class_name& Class_name::operator=(const Class_name&);
- StringBad& StringBad::operator=(const StringBad&);
- 호출되는 예
  - knot = headline1;
  - StringBad metoo = knot;
- 문제 해결 방법
  - 자기 자신에게 대입하는 바보짓 처리
  - 왼쪽 객체가 가지고 있는 이전에 할당된 데이터를 해제
  - 오른쪽 객체의 값을 깊게 복사
  - 호출한 왼쪽 객체에 대한 참조를 리턴
- 보충 (738쪽)
  - StringBad& StringBad::operator=(const char *);

개선된 String 클래스

- friend bool operator<(const String&, const String&);
- friend bool operator>(const String&, const String&);
- friend bool operator==(const String&, const String&);
- friend istream& operator>>(istream&, String&);
- char& operator[](int i);
- const char& operator[](int i); const

비교 멤버 (734쪽)

```
friend bool operator<(const String& s1, const
   String& s2)
{
  return (std::strcmp(s1.str,s2.str)<0);
}
```

- 프렌드 함수로 String 객체와 C 스타일 문자열 비교
  - if (“love”==answer)
  - → if (operator==(“love”,answer)) // 프렌드 함수
  - → if (operator==(String(“love”),answer)) // 생성자

[] 표기로 개별 문자 접근 (735쪽)

- String opera("The Magic Flute");
- opera[4]
  - opera[](int i)
  - opera.operator[](4)
- means[0] = 'r'; // const 가 아닌 operator
  - means.operator[](0) = 'r';
  - means.str[0] = 'r';
- const 객체를 위해 const 멤버 함수로 오버로딩

static 클래스 멤버 함수

- 함수 정의와 함수 선언이 분리되어 있으면, static은 함수 선언에 나타남
- 객체에 의해 호출되지 않음
- this 포인터도 없음
- 클래스 이름과 사용 범위 연산자(::)로 호출
  - int count = String::HowMany();
- static 데이터 멤버만 사용 가능, 아니면 내부적으로 객체 생성

생성자에서 new를 사용할 경우

- 파괴자에서도 delete를 사용
- new는 delete로, new[]는 delete[]로
- 포인터를 NULL 또는 0으로 초기화 허용
- 깊은 복사를 하는 복사 생성자
  - EC++ #12
- 깊은 복사를 하는 대입 연산자
  - EC++ #11

객체 리턴

- 객체, 객체 참조, const 객체, const 객체 참조
- const 객체 참조
  - 효율성 (복사 없음)
  - 실행 중에 존재하는 객체에 대한 참조
- 객체 참조
  - 효율성(복사 없음)과 필요성
  - 연쇄적인 대입이나 출력
- 객체 리턴
  - 복사 생성자가 사용되며, 리턴되는 객체가 로컬일 때 유용
- const 객체 리턴
  - 객체 리턴이나 참조 리턴의 오남용(752쪽)을 막음
    - net = force1+force2; // 오케이
    - force1+force2 = net; // 이상한 프로그래밍
    - cout << (force1+force2 = net).magval() << endl; // 이상한 프로그래밍

new에 의한 객체의 초기화(755쪽)

- Class_name: 클래스 이름, value가 Type_name 형
  - Class_name* pC = new Class_name(value);
    - 다음 생성자를 호출
      - Class_name(Type_name);
    - 다음 생성자도 가능 (사소한 변환)
      - Class_name(const Type_name &);
- 모호성이 없는 한 일반적인 변환은 가능
  - 예: int 를 double 로
- 디폴트 생성자 호출
  - Class_name* pC = new Class_name

포인터와 객체

- 파괴자
  - 자동 변수 – 블록을 벗어나면 호출
  - 정적 변수 – 프로그램이 종료될 때 호출
  - 동적 변수 – new 로 생성 / delete 될 때 호출
- new로 포인터 초기화
  - String* f = new String(myString);
- 디폴트 생성자
  - String* g = new String;
- String(const char *) 생성자
  - String* f = new String("my my my");
- 멤버에 접근하려면 ->, 객체에 접근하려면 *

위치 지정 new 다시 보기

- 메모리 지정 new, Placement new
- placenew1.cpp (760쪽)
  - 동일한 위치에 위치 지정 new를 두 번 한 것
    - 첫 번째 위치 지정 new의 파괴자는 호출되지 않을 것임
    - 생성자나 메소드가 동적 메모리 할당을 한다면 더 상황이 나빠짐
  - 위치 지정 new로 생성된 객체들은 파괴자를 호출하지 않음
- 메모리가 중복되지 않게 한다
  - p1 = new (buffer) JustTesting;
  - p2 = new (buffer+sizeof(JustTesting)) JustTesting;
- 파괴자를 명시적으로 호출한다
  - p1->~ JustTesting();
  - p3->~ JustTesting();

큐 시뮬레이션

- 인터페이스 – 769쪽 코드
- 링크드 리스트 – 770쪽
- 큐 클래스 선언(771쪽) – 내포된 구조체 정의로 링크드 리스트를 넣음 (772쪽 노트)
- const 멤버 변수 초기화(773쪽) – 멤버 초기자 리스트 문법 774쪽)
- enqueue(775쪽), dequeue(777쪽), 파괴자 (779쪽)
- 새로운 고객 함수 newcustomer (785쪽)