dongguk UNIVERSITY

심화프로그래밍

실습강의 5주차

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. :김동국19919876실습5.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**

인라인 함수

- **인라인 함수(inline function)**
  - inline 키워드를 통해 인라인 함수를 사용 가능하다
  - 프로그램의 **성능 개선과 오버헤드**를 줄이기 위해 사용된다 (프로그램 빠르게 실행)
  - 인라인 함수를 사용하면 다음과 같은 많은 **이점**을 얻을 수 있다
    - 함수 내부의 코드를 재사용할 수 있다.
    - 인스턴트 코드보다 함수에서 코드를 변경하거나 업데이트하기가 더 쉽다.
    - 함수 이름을 통해 코드가 무엇을 의미하는지 이해하기 더 쉽다.
    - 함수 호출 인수가 함수 매개 변수와 일치하는지 확인하기 위해 타입 검사를 한다.
    - 함수는 프로그램을 디버그하기 쉽게 만든다.
  - 내부 루프가 없는 **짧은 함수**에 사용하기 적합하다
  - 컴파일러는 인라인에 대한 요청을 자유롭게 무시할 수 있다.
    - 개발자가 인라인으로 작성하지 않더라도 똑똑한 현대의 컴파일러는 코드 최적화를 한다

객체 포인터

- **객체 이름으로 접근하기**
  - 클래스명 객체명
    - 예1) Student aClass();
    - 예2) Student aClass(30);
  - 클래스의 멤버를 접근하기 위해서는 .를 사용함
    - 예) aClass.getNum()
- **객체 포인터로 접근하기**
  - 클래스명 *객체명

    예) Student *s;
  - C언어의 포인터가 값의 주소값을 저장하듯, **객체의 주소 값**을 가질 수 있음
    - 예) s = &aClass;
  - 포인터로 클래스의 멤버를 접근하기 위해서는 ->를 사용함
    - 객체포인터->멤버
    - 예1) s->getNum()
    - 예2) (*s).getNum()

객체 배열 생성 및 소멸

- **객체 배열 생성**
  - C언어의 기본 데이터 타입 배열을 선언한 방식과 동일하게 객체 배열 사용 가능
    - int num[5]; // 정수형 배열 선언
    - Student aClass[5]; // Student 클래스 타입의 배열 선언
  - 객체 배열을 위한 **공간 할당**이 이루어지며, 각 원소의 **객체마다 생성자 실행**됨
    - aClass[0], aClass[1], aClass[2] … → 매개 변수가 없는 생성자 호출
  - 매개 변수가 있는 생성자
    - Student aClass[5](10) // 컴파일 오류 발생
- **객체 배열 소멸**
  - 각 원소의 객체마다 생성된 **순으로 소멸자가 실행**됨
    - … aClass[2], aClass[1], aClass[0]

객체 배열 생성 및 소멸 예제 [1/2]

- 객체 배열을 이해하기 위한 예제 1 – 객체 이름으로 접근하기

```
1. class TestResult {
2. private:
3.     int myScore;
4.     int myNumber;
5.     string myName;
6. public:
7.     TestResult() {
8.         myScore = 0;
9.         myNumber = 0;
10.        myName = "none";
11.    }
12.    void setInfo(int score, int number, string name);
13.    void printAll();
14.};
15.void TestResult::setInfo(int score, int number, string name) {
16.    myScore = score;
17.    myNumber = number;
18.    myName = name;
19.}
20.void TestResult::printAll() {
21.    cout << myNumber << '\n';
22.    cout << myName << '\n';
23.    cout << myScore << '\n';
24.}
```

객체 배열 생성 및 소멸 예제 [2/2]

- 객체 배열을 이해하기 위한 예제 1 – 객체 이름으로 접근하기

```
1.  int main() {
2.      TestResult student[3];
3.      int num, score;
4.      string name;
5.
6.      for (int i = 0; i < 3; i++) {
7.          cout << "학번 : ";
8.          cin >> num;
9.          cout << "이름 : ";
10.         cin >> name;
11.         cout << "점수 : ";
12.         cin >> score;
13.         student[i].setInfo(score, num, name);
14.     }
15.
16.     student[0].printAll();
17.     student[1].printAll();
18.     student[2].printAll();
19.     return 0;
20. }
```

```
학번 : 2022123456
이름 : 홍길동
점수 : 80
학번 : 2022654321
이름 : 김철수
점수 : 90
학번 : 2022987456
이름 : 이유리
점수 : 95
2022123456
홍길동
80
2022654321
김철수
90
2022987456
이유리
95
```

객체 배열 초기화 [1/2]

- **객체 배열 초기화 방법**
  - 클래스명 객체명[3] = { 클래스명(), 클래스명(11), 클래스명(22) };
    - 예) Student aClass[3] = { Student(), Student(11), Student(22) };
  - aClass[0] 객체가 생성될 때 Student() 생성자 호출
    - 초기화를 진행하지 않을 경우, 디폴트 생성자 실행
  - aClass[1] 객체가 생성될 때 Student(11) 생성자 호출
  - aClass[2] 객체가 생성될 때 Student(22) 생성자 호출
    - 매개 변수가 있는 경우 데이터 타입과 맞춰서 초기화 필요

객체 배열 초기화 예제 [2/2]

- Student 클래스, 객체 배열 초기화

```
1.  class Student {
2.  private:
3.      int num;
4.  public:
5.      Student() { num = 123456; }
6.      Student(int n) { num = n; }
7.      int getNum() { return num; }
8.  };
9.
10. int main() {
11.     Student aClass[3] = {Student(), Student(11), Student(22)};
12.     for (int i = 0; i < 3; i++) {
13.         cout << aClass[i].getNum() << endl;
14.     }
15.     return 0;
16. }
```

```
123456
11
22
```

2차원 배열

- **2차원 배열 사용**
  - 클래스명 객체[n][m];
    - 예1) Student aClass[2][4] = { Student(111), Student(222), Student(333), Student(444) };

```
1. class Student {
2. private:
3.     int num;
4. public:
5.     Student() { num = 123456; }
6.     Student(int n) { num = n; }
7.     void setNum(int n) { num = n; }
8.     int getNum() { return num; }
9. };
10.
11.int main() {
12.    Student aClass[2][4] = { Student(111), Student(222), Student(333), Student(444) };
13.
14.    aClass[1][0].setNum(55);
15.    aClass[1][1].setNum(66);
16.    aClass[1][2].setNum(77);
17.    aClass[1][3].setNum(88);
18.
19.    for (int i = 0; i < 2; i++) {
20.        for(int j = 0; j < 4; j++ )
21.            cout << aClass[i][j].getNum() << endl;
22.    }
23.}
```

2차원 배열 초기화 **생성자 활용**

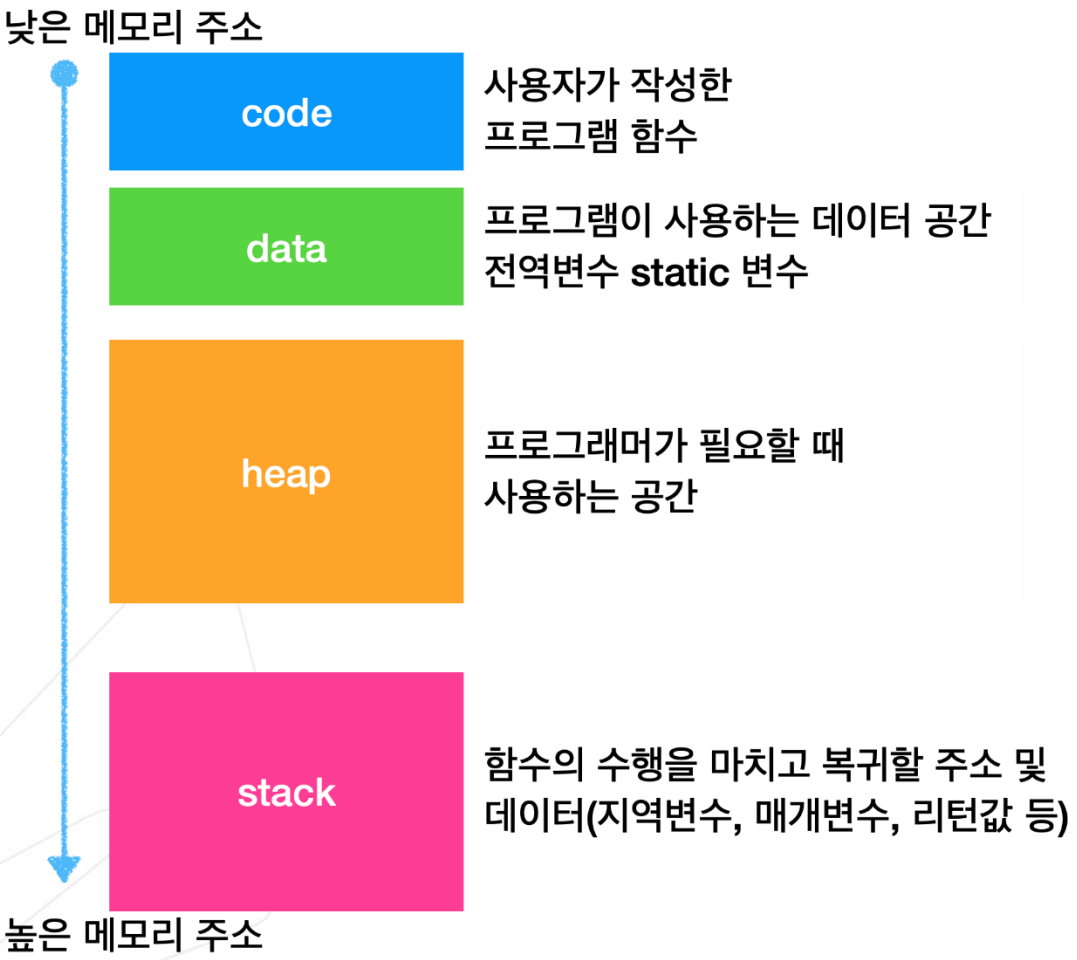
→ 2차원 배열 초기화 **멤버함수 활용**

동적 메모리 할당 및 반환 [1/3]

- **메모리 할당**
  - 정적 할당
    - 변수 선언을 통해 필요한 메모리 할당
  - 동적 할당
    - 필요한 양이 예측되지 않을 경우, 프로그램 작성 시 할당 받을 수 없음
    - 프로그래머가 필요할 때 힙 메모리로부터 할당 받을 수 있음
    - C언어의 경우, 할당: malloc(), 해제: free()
  - new 연산자
    - **기본타입 메모리, 배열, 객체, 객체배열 할당**
    - 객체의 동적 생성: 힙 메모리로부터 객체를 위한 **메모리 할당 요청 후 생성자 호출**
  - delete 연산자
    - new로 할당받은 **메모리 반환**
    - 객체의 동적 소멸: **소멸자 호출 후 객체를 힙에 메모리 반환**

동적 메모리 할당 및 반환 [2/3]

- 메모리 구조



출처: https://zangzangs.tistory.com/107

동적 메모리 할당 및 반환 [3/3]

- **동적 할당 및 반환 연산자**
  - 타입 *변수명 = new 타입(초기화1, 초기화2, 초기화3);
  - delete 변수명, delete[] 변수명

```
1.  int main() {
2.      int* point = new int; // 메모리 동적할당
3.      *point = 100;
4.      cout << *point;
5.      delete point; // 메모리 할당해제
6.  }
```

```
1.  int main() {
2.      int* point = new int[5]; // 배열 형태 메모리 동적할당
3.
4.      for (int i = 0; i < 5; i++) {
5.          point[i] = i;
6.      }
7.      for (int i = 0; i < 5; i++) {
8.          cout << point[i] << endl;
9.      }
10.
11.     delete[] point; // 배열 형태 메모리 할당해제
12. }
```

this 포인터 [1/4]

- **this 포인터**
  - 클래스의 멤버 함수를 호출할 때 C++은 어떻게 호출할 객체(인스턴스)를 찾는가?
    - **this** 라는 숨겨진 포인터를 사용하여 찾을 수 있음
  - C++에서 this란 자기 자신을 나타내는 말
  - this 포인터는 클래스의 실제 인스턴스에 대한 주소를 가리키는 포인터
  - 컴파일러에 의해 첫 번째 매개변수로 전달된 인스턴스 주소는 this 포인터 변수에 저장됨
- Student 클래스 예시

```
1.  class Student {
2.  private:
3.      int age;
4.  public:
5.      Student(int num) {
6.          age = num;
7.      }
8.      void print_student() {
9.          cout << age << "살" << endl;
10.     }
11. };
```

```
1.  int main(void) {
2.      Student s1(22);
3.      s1.print_student();
4.      return 0;
5.  }
```

22살

st1.print_student(&s1)

this 포인터 [2/4]

- **this 포인터는 언제 사용할까? – 명시적 참조**
  - 클래스의 멤버 변수(생성자)와 매개 변수가 동일할 때 사용할 수 있다
  - 객체 자신의 주소를 리턴할 때 사용할 수 있다

```
1.  class Student {
2.  private:
3.      int age;
4.  public:
5.      Student(int age) {
6.          this->age = age;
7.      }
8.      void print_student() {
9.          cout << age << "살" << endl;
10.     }
11. };
```

멤버 변수 age와 매개 변수 age가 같을 때,
this->age는 멤버 변수를 뜻하고 age는 매개 변수를 뜻함

```
1.  class Student {
2.  public:
3.      void print_student() {
4.          cout << this << endl;
5.      }
6.  };
7.
8.  int main(void) {
9.      Student s2;
10.     s2.print_student();
11.     cout << &s2 << endl;
12.     return 0;
13. }
```

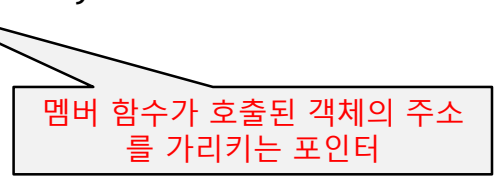
```
012FFE97
012FFE97
```

this를 이용하여 출력한 값과 &s2의 값은 같음
즉, this는 자기 자신을 가리키는 것

this 포인터 예제 [3/4]

- Student 클래스 this 포인터 사용 예제 – 두 가지 객체 선언

```
1.  class Student {
2.  public:
3.      void print_student() {
4.          cout << this << endl;
5.      }
6.  };
7.
8.  int main(void) {
9.      Student s2;
10.     s2.print_student();
11.     cout << "&s2: " << &s2 << endl;
12.
13.     Student s3;
14.     s3.print_student();
15.     cout << "&s3: " << &s3 << endl;
16.     return 0;
17. }
```




```
0099F923
&s2: 0099F923
0099F917
&s3: 0099F917
```

this 포인터 예제 [4/4]

- Calc 클래스 this 포인터 사용 예제 – 멤버 함수 연속 호출

```
1.  class Calc {
2.  private:
3.      int m_Value = 0;
4.  public:
5.      Calc& Add(int value) { m_Value += value; return *this; }
6.      Calc& Sub(int value) { m_Value -= value; return *this; }
7.      Calc& Mul(int value) { m_Value *= value; return *this; }
8.      int GetValue() { return m_Value; }
9.  };
10.
11. int main() {
12.     Calc calc;
13.     calc.Add(5).Sub(3).Mul(4);
14.     1. (calc.Add(5)).Sub(3).Mul(4); // m_Value = 0 + 5
15.     2. (calc.Sub(3)).Mul(4);        // m_Value = 5 - 3
16.     3. calc.Mul(4);*/               // m_Value = 2 * 4; => 8
17.     cout << calc.GetValue() << endl; // 8
18.     return 0;
19. }
```

```
8
```

this 포인터 적용하기

- Simple 클래스 this 포인터 사용 예제 – 매개 변수 포함

```
1.  class Simple {
2.  private:
3.      int m_ID;
4.  public:
5.      Simple(int id) {
6.          SetID(id);
7.      }
8.
9.      void SetID(int id) {
10.         m_ID = id;
11.     }
12.
13.     int GetID() {
14.         return m_ID;
15.     }
16. };
```

```
1.  int main() {
2.      Simple simple(159);
3.      simple.SetID(123456789);
4.      cout << simple.GetID() << endl;
5.      return 0;
6.  }
```

```
123456789
```

string 클래스를 이용한 문자열 사용 [1/4]

- **string 클래스**
  - C++ STL(Standard Template Library)에서 제공하는 클래스로 문자열을 다루는 클래스
  - C에서 char* 또는 char[]의 형태로 문자열을 다뤘지만, C++에서는 하나의 변수 type처럼 사용함
  - C에서 문자열의 끝에 '₩0' 문자를 넣어 표현했지만, C++에서는 '₩0' 없이 문자열의 끝을 알 수 있음
    - string 관련 함수로 length() 또는 size()를 통해 실제 문자열의 길이나 사이즈를 가져올 수 있음
  - append() 함수로 문자열의 끝에 새로운 문자열을 추가할 수 있다
    - str1.append("새로운 문자열");
    - str2.append("새로운 문자열, 시작 번호, 개수);
  - find() 함수로 원하는 문자열의 시작 위치를 반환하여 알아낼 수 있다.
    - str1.find(찾을 문자열);
    - str2.find(찾을 문자);
    - str3.find(찾을 문자열, 시작 위치);
  - compare() 함수는 두 문자열 간의 내용을 비교할 수 있다.
    - str1.compare(비교할 문자열);
  - replace() 함수는 특정 문자열을 다른 문자열로 대체할 수 있다.
    - str1.replace(변경할 문자열 시작 위치, 새로운 문자열의 길이, 새로운 문자열);
  - …

string 클래스를 이용한 문자열 사용 [2/4]

- **string 클래스 사용, length()와 size() 함수 사용**
  - 일반 변수처럼 선언과 동시에 초기화 가능하며 변수끼리의 대입도 가능함
  - length() 함수는 문자열의 길이를 반환, size() 함수는 객체의 메모리에서 사용하는 크기를 반환함
    - length()의 경우 널문자(₩0)까지의 길이가 아닌 실제 문자열의 길이만 반환함

```
1.  int main(void) {
2.      string str1 = "Hello, ";
3.      string str2 = "my name is dongguk";
4.      string str3;
5.
6.      str3 = str1 + str2;
7.      cout << str3 << endl;
8.
9.      str1 += str2;
10.     cout << str1 << endl;
11. }
```

string() 클래스 사용 예시

```
Hello, my name is dongguk
Hello, my name is dongguk
```

```
1. int main(void) {
2.     string str1 = "Hello, ";
3.     string str2 = "my name is dongguk";
4.     string str3;
5.
6.     str3 = str1 + str2;
7.     cout << str3.length() << endl;
8.     cout << str3.size() << endl;
9. }
```

length()와 size() 함수 사용 예시

```
25
25
```

string 클래스를 이용한 문자열 사용 [3/4]

- **string 클래스 사용, append()와 find() 함수 사용**

```
1. int main(void) {
2.     string str1 = "Hello, ";
3.     string str2 = "AABBCCDDEEFF";
4.
5.     cout << str1.append("Bye!") << endl;
6.     cout << str2.append("Hello,", 1, 3) << endl;
7. }
```

```
Hello, Bye!
AABBCCDDEEFFell
```

append() 함수 사용 예시

```
1. int main(void) {
2.     string str1 = "Hello";
3.
4.     cout << str1.find("lo") << endl;
5.     cout << str1.find('l') << endl;
6.     cout << str1.find('o', 3) << endl;
7. }
```

```
3
2
4
```

find() 함수 사용 예시

string 클래스를 이용한 문자열 사용 [4/4]

- **string 클래스 사용, compare()와 replace() 함수 사용**

```
1.  int main(void) {
2.      string str1 = "Hello";
3.      string str2 = "my name is dongguk";
4.
5.      if (str1.compare(str2) == 0) {
6.          cout << "두 문자열이 일치합니다." << endl;
7.      }
8.      else if (str1.compare(str2) < 0) {
9.          cout << str1 << "는 사전 순으로 앞입니다." << endl;
10.     }
11.     else {
12.         cout << str1 << "는 사전 순으로 뒤입니다." << endl;
13.     }
14. }
```

Hello는 사전 순으로 앞입니다.

compare() 함수 사용 예시

```
int main(void) {
    string str1 = "Hello, dongguk";
    string str2 = "Hello";
    string str3 = "Hiooo";
    int index;

    index = str1.find(str2);
    cout << str1.replace(index, str3.length(), str3) << endl;
}
```

Hiooo, dongguk

replace() 함수 사용 예시

string 클래스의 멤버 함수들 [1/3]

- **String 클래스의 다양한 멤버 함수들 1**

string의 특정 원소 접근	
str.at(index)	Index 위치의 문자 반환, 유효한 범위인지 체크 O
str[index]	Index 위치의 문자 반환. 유효한 범위인지 체크 X. at 함수보다 접근이 빠름
str.front()	문자열의 가장 앞 문자 반환
str.back()	문자열의 가장 뒤 문자 반환

string의 크기	
str.length()	문자열 길이 반환
str.size()	문자열 길이 반환 (length와 동일)
str.capacity()	문자열이 사용중인 메모리 크기 반환
str.resize(n)	string을 n의 크기로 만듦. 기존의 문자열 길이보다 n이 작다면 남은 부분은 삭제하고, n이 크다면 빈공간으로
str.resize(n, 'a')	n이 string의 길이보다 더 크다면, 빈 공간을 'a'로 채움
str.shrink_to_fit()	string의 capacity가 실제 사용하는 메모리보다 큰 경우 낭비되는 메모리가 없도록 메모리를 줄여줌
str.reserve(n)	size = n만큼의 메모리를 미리 할당해줌
str.empty()	str이 빈 문자열인지 확인

string 클래스의 멤버 함수들 [2/3]

- **String 클래스의 다양한 멤버 함수들 2**

string에 삽입, 추가, 삭제	
str.append(str2)	str 뒤에 str2 문자열을 이어 붙여줌 ('+' 와 같은 역할)
str.append(str2, n, m)	str 뒤에 'str2의 n index부터 m개의 문자'를 이어 붙여줌
str.append(n, 'a')	str 뒤에 n개의 'a'를 이어 붙여줌
str.insert(n, str2)	n번째 index 앞에 str2 문자열을 삽입함.
str.replace(n, k, str2)	n번째 index부터 k개의 문자를 str2로 대체함
str.clear()	저장된 문자열을 모두 지움
str.erase(n, m)	n번째 index부터 m개의 문자를 지움
str.erase(n, m) (iterator)	n~m index의 문자열을 지움 (n과 m은 iterator)
str.erase()	clear와 같은 동작
str.push_back(c)	str의 맨 뒤에 c 문자를 붙여줌
str.pop_back()	str의 맨 뒤의 문자를 제거
str.assign(str2)	str에 str2 문자열을 할당. (변수 정의와 동일)

string 클래스의 멤버 함수들 [3/3]

- **String 클래스의 다양한 멤버 함수들 3**

기타 유용한 string 멤버 함수	
str.find("abcd")	"abcd"가 str에 포함되어있는지를 확인. 찾으면 해당 부분의 첫번째 index를 반환
str.find("abcd", n)	n번째 index부터 "abcd"를 find
str.substr()	str 전체를 반환
str.substr(n)	str의 n번째 index부터 끝까지의 문자를 부분문자열로 반환
str.substr(n, k)	str의 n번째 index부터 k개의 문자를 부분문자열로 반환
str.compare(str2)	str과 str2가 같은지를 비교. 같다면 0, str<str2 인 경우 음수, str>str2 인 경우 양수를 반환
swap(str1, str2)	str1과 str2를 바꿔줌. reference를 교환하는 방식
isdigit(c)	c 문자가 숫자이면 true, 아니면 false를 반환
isalpha(c)	c 문자가 영어이면 true, 아니면 false를 반환
toupper(c)	c 문자를 대문자로 변환
tolower(c)	c 문자를 소문자로 변환

연습문제(4장)

* 1~5번 문제에 사용되는 Rect 클래스. Rect 클래스는 폭과 높이로 사각형을 추상화한다.

```
class Rect {
   int width, height;
public:
   Rect(int w, int h) { width = w; height = h; }
   int getWidth() { return width; }
   int getHeight() { return height; }
   int getArea();
};

int Rect::getArea() {
   return width*height;
}
```

연습문제(4장)

1. Rect의 객체를 다루는 다음 코드를 작성하려고 한다. 아래의 문제에 따라 빈칸에 적절한 코드를 삽입하라.

```
int main() {
   Rect r(2,3);
   _____(1)
   _____(2)
   _____(3)
}
```

(1) Rect 클래스에 대한 포인터 변수 p를 선언하라.
(2) 선언된 포인터 변수 p에 객체 r의 주소를 지정하라.
(3) 포인터 변수 p를 이용하여 객체 r의 폭과 높이를 출력하라.

연습문제(4장)

2. 사용자로부터 폭과 높이 값을 입력받아 동적으로 Rect 객체를 생성하고 면적을 구하여 출력하는 코드를 작성하고자 한다. 다음 물음에 따라 빈칸을 채워라.

```
int main() {
   Rect *q;
   int w, h;
   cin >> w >> h; // 사용자로부터 사각형의 폭과 높이를 w, h에 각각 입력받는다.

   _____(1)
   _____(2)
   _____(3)
}
```

(1) 포인터 변수 q에 wxh 크기의 사각형을 표현하는 Rect 객체를 동적으로 생성한다.
(2) 포인터 q를 이용하여 사각형의 면적을 출력한다.
(3) 생성한 객체를 반환한다.

연습문제(4장)

3. Rect 객체나 배열을 생성하는 다음 코드 중 컴파일 오류가 발생하는 것은?

① Rect a;
② Rect b(5, 6);
③ Rect c[2] = { Rect(1, 1), Rect(2, 3) };
④ Rect d[2][3] = { {Rect(1,2), Rect(2,3), Rect(3,4)}, {Rect(1,1), Rect(2,2), Rect(3,3)} };

4. Rect 객체의 배열을 생성하는 다음 코드는 컴파일 오류가 발생한다. 컴파일 오류가 발생하지 않기 위해 Rect 클래스를 어떻게 수정하여야 하는가?

```
Rect *p = new Rect[10];
```

5. Rect 클래스에 다음과 같은 기본 생성자를 삽입하고,

```
Rect() { width = 1; height = 1; }
```

다음 배열 r 생성 후, 배열 r의 사각형 면적의 합을 출력하는 코드를 작성하라.

```
Rect r[5] = { Rect(), Rect(2, 3), Rect(3,4), Rect(4,5), Rect(5,6) };
```

연습문제(4장)

6. public 속성의 getVolume() 멤버 함수를 가진 Cube 클래스에 대해, 다음 코드가 있다.

```
Cube c;
Cube *p = &c;
```

다음 중 컴파일 오류가 발생하는 것은?

① c.getVolume();
② p->getVolume();
③ (*p).getVolume();
④ c->getVolume();

7. 다음 객체 배열에 관해 잘못 설명된 것은?

```
Cube c[4];
```

① 배열 c가 생성될 때 c[0], c[1], c[2], c[3]의 4개의 Cube 객체가 생성된다.
② 기본 생성자 Cube()가 4번 호출된다.
③ 배열 c가 소멸될 때 c[3], c[2], c[1], c[0]의 순서로 소멸자가 실행된다.
④ delete c; 코드로 배열 c를 소멸한다.

연습문제(4장)

8. 다음 프로그램이 실행될 때 출력되는 결과는 무엇인가?

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

class Color {
   string c;
public:
   Color() { c = "white"; cout << "기본생성자" << endl; }
   Color(string c) { this->c = c; cout << "매개변수생성자" << endl; }
   ~Color() { cout << "소멸자" << endl; }
};

class Palette {
   Color *p;
public:
   Palette() { p = new Color[3]; }
   ~Palette() { delete [] p; }
};

int main() {
   Palette *p = new Palette();
   delete p;
}
```

연습문제(4장)

9. new와 delete는 무엇인가?

① C++의 기본 연산자
② C++ 표준 함수
③ C++의 표준 객체
④ C++의 특수 매크로

10. 다음 코드의 문제점은 무엇인가?

```
Cube *p = new Cube [4];
delete p;
```

11. this에 대해 잘못 말한 것은?

① this는 포인터이다.
② this는 컴파일러에 의해 묵시적으로 전달되는 매개 변수이다.
③ this는 static 함수를 포함하여 멤버 함수 내에서만 다루어지는 객체 자신에 대한 포인터이다.
④ 연산자 중복에서 this가 필요하다.

12. this의 활용에 대해 잘못 설명한 것은?

① this는 클래스의 멤버 함수 외의 다른 함수에서는 사용할 수 없다.
② this는 static 멤버 함수에는 사용할 수 없다.
③ this는 생성자에서 사용할 수 없다.
④ 어떤 멤버 함수에서는 this를 리턴하기도 한다.

연습문제(4장)

13. this를 최대한 많이 활용하여 다음 클래스를 가장 바람직하게 수정하라.

```
class Location {
   int width, height;
public:
   Location() { width = height = 0; }
   Location(int w, int h) {
      width = w; height = h;
   }
   void show();
};
void Location::show() {
   cout << width << height << endl;
}
```

14. 메모리 누수란 어떤 상황에서 발생하는가?

연습문제(4장)

15. 함수 f()가 실행되고 난 뒤 메모리 누수가 발생하는지 판단하고 메모리 누수가 발생하면 발생하지 않도록 수정하라.

(1)

```
void f() {
   char *p = new char [10];
   strcpy(p, "abc");
}
```

(2)

```
void f() {
   int *p = new int;
   int *q = p;
   delete q;
}
```

(3)

```
int f() {
   int n[10] = { 0 };
   return n[0];
}
```

(4)

```
void f() {
   int *p;
   for(int i=0; i<5; i++) {
      p = new int;
      cin >> *p;
      if(*p % 2 == 1) break;
   }
   delete p;
}
```

연습문제(4장)

16. string 클래스를 사용하기 위해 필요한 헤더 파일은 무엇인가?

① <string> ② <string.h> ③ <cstring> ④ <iostream>

17. string s1 = "123"; string s2 = "246"; 일 때, a와 b의 문자열 속에 있는 수를 더하여 369를 출력하고자 한다. 아래 빈칸을 채워라.

```
int n = _____ (s1);
int m = _____ (s2);
cout << n + m;
```

18. 문자열을 다루고자 한다. C-스트링과 string 클래스에 대해 설명이 틀린 것은?

① C-스트링은 문자의 배열을 이용하여 문자열을 표현한다.

② string 클래스가 문자열을 객체화하므로 C-스트링보다 사용하기 쉽다.

③ string 클래스가 좋기는 하지만 C++의 표준이 아니므로 가급적 사용하지 않는 것이 좋다.

④ string 클래스는 문자열만 다루지 대문자를 소문자로 변환하는 등 문자를 조작하는 기능은 없다.

연습문제(4장)

19. 다음 프로그램의 각 라인을 string 클래스에서 제공하는 연산자를 이용하여 고쳐라.

```
string a("My name is Jane.");
char ch = a.at(2);
if(a.compare("My name is John.") == 0) cout << "same";
a.append("~~");
a.replace(1, 1, "Y");
```

실습과제(4장)

1. 다음은 색의 3요소인 red, green, blue로 색을 추상화한 Color 클래스를 선언하고 활용하는 코드이다. 빈칸을 채워라. red, green, blue는 0~255의 값만 가진다.

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

class Color {
    int red, green, blue;
Public:
    Color() { red = green = blue = 0; }
    Color(int r, int g, int b) { red = r; green = g; blue = b; }
    void setColor(int r, int g, int b) { red = r; green = g; blue = b; }
    void show() { cout << red << ' ' << green << ' ' << blue << endl; }
};

int main() {
    Color screenColor(255, 0, 0); // 빨간색의 screenColor 객체 생성
    Color *p;                     // Color 타입의 포인터 변수 p 선언
    ____________________          // (1) p가 screenColor의 주소를 가지도록 코드 작성
    ____________________          // (2) p와 show()를 이용하여 screenColor 색 출력
    ____________________          // (3) Color의 일차원 배열 colors 선언. 원소는 3개
    ____________________          // (4) p가 colors 배열을 가리키도록 코드 작성
```

실습과제(4장)

1. 다음은 색의 3요소인 red, green, blue로 색을 추상화한 Color 클래스를 선언하고 활용하는 코드이다. 빈칸을 채워라. red, green, blue는 0~255의 값만 가진다.

```
(위 페이지 이어서)
    // (5) p와 setColor()를 이용하여 colors[0], colors[1], colors[2]가
    // 각각 빨강, 초록, 파랑색을 가지도록 코드 작성
    _____________________
    _____________________
    _____________________

    // (6) p와 show()를 이용하여 colors 배열의 모든 객체의 색 출력. for 문 이용
    _____________________
    _____________________
    _____________________
}
```

```
255    0     0
255    0     0
0     255    0
0      0    255
```

실습과제(4장)

2. 정수 공간 5개를 배열로 동적 할당받고, 정수를 5개 입력받아 평균을 구하고 출력한 뒤 배열을 소멸시키도록 main() 함수를 작성하라.

```
정수 5개 입력 >> 1  2  4  5  10
평균 4.4
```

3. string 클래스를 이용하여 빈칸을 포함하는 문자열을 입력받고 문자열에서 'a'가 몇 개 있는지 출력하는 프로그램을 작성하라.

```
문자열 입력 >> Are you happy? I am so happy.
문자 a는 3개 있습니다.
```

1) 문자열에서 'a'를 찾기 위해 string 클래스의 멤버 at() 나 []를 이용하여 작성하라.
2) 문자열에서 'a'를 찾기 위해 string 클래스의 find() 멤버 함수를 이용하여 작성하라. text.find('a', index);는 text 문자열의 index위치부터 'a'를 찾아 문자열 내 인덱스를 리턴한다.

실습과제(4장)

4. 다음과 같은 Sample 클래스가 있다.

```
class Sample {
    int *p;
    int size;
public:
    Sample(int n) {  // 생성자
        size = n; p = new int [n]; // n개 정수 배열의 동적 생성
    }
    void read(); // 동적 할당받은 정수 배열 p에 사용자로부터 정수를 입력 받음
    void write(); // 정수 배열을 화면에 출력
    int big(); // 정수 배열에서 가장 큰 수 리턴
    ~Sample(); // 소멸자
};
```

다음 main() 함수가 실행되도록 Sample 클래스를 완성하라.

```
int main() {
    Sample s(10); // 10개 정수 배열을 가진 Sample 객체 생성
    s.read(); // 키보드에서 정수 배열 읽기
    s.write(); // 정수 배열 출력
    cout <<"가장 큰 수는 " << s.big() << endl; // 가장 큰 수 출력
}
```

```
100  4  -2  9  55  300  44  38  99  -500
100  4  -2  9  55  300  44  38  99  -500
가장 큰 수는 300
```

실습과제(4장)

5. string 클래스를 이용하여 사용자가 입력한 영문 한 줄을 입력받고 글자 하나만 랜덤하게 수정하여 출력하는 프로그램을 작성하라.

```
아래에 한 줄을 입력하세요. (exit를 입력하면 종료합니다)
>>Falling in love with you.
Falling in love wxth you.
>>hello world
hello wobld
››exit
```

랜덤 정수를 발생시키기 위해 다음 두 라인의 코드가 필요하며, <cstdlib>와 <ctime> 헤더 파일을 include 해야 한다.

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

실습과제(4장)

6. string 클래스를 이용하여 사용자가 입력한 영문 한 줄을 문자열로 입력받고 거꾸로 출력하는 프로그램을 작성하라.

```
아래에 한 줄을 입력하세요. (exit를 입력하면 종료합니다)
››Delicious C++
++C suoicileD
»›I love programming.
.gnimmargorp evol I
>>exit
```

7. 다음과 같이 원을 추상화한 Circle 클래스가 있다. Circle 클래스와 main() 함수를 작성하고 3개의 Circle 객체를 가진 배열을 선언하고, 반지름 값을 입력받고 면적이 100보 다 큰 원의 개수를 출력하는 프로그램을 완성하라. Circle 클래스도 완성하라.

```
class Circle {
    int radius;  // 원의 반지름 값
public:
    void setRadius (int radius);  // 반지름을 설정한다.
    double getArea(); // 면적을 리턴한다.
};
```

```
원 1의 반지름 >> 5
원 2의 반지름 >> 6
원 3의 반지름 >> 7
면적이 100보다 큰 원은 2개 입니다
```

실습과제(4장)

8. 실습 문제 7의 문제를 수정해보자. 사용자로부터 다음과 같이 원의 개수를 입력받고, 원의 개수만큼 반지름을 입력받는 방식으로 수정하라. 원의 개수에 따라 동적으로 배 열을 할당받아야 한다.

```
원의 개수 >> 4
원 1의 반지름 >> 5
원 2의 반지름 >> 6
원 3의 반지름 >> 7
원 4의 반지름 >> 8
면적이 100보다 큰 원은 3개 입니다
```

실습과제(4장)

9. 다음과 같은 Person 클래스가 있다. Person 클래스와 main() 함수를 작성하여, 3개 의 Person 객체를 가지는 배열을 선언하고, 다음과 같이 키보드에서 이름과 전화번호를 입력받아 출력하고 검색하는 프로그램을 완성하라.

```
class Person {
    string name;
    string tel;
public:
    Person();
    string getName () { return name; }
    string getTel() { return tel; }
    void set(string name, string tel);
};
```

```
이름과 전화 번호를 입력해 주세요
사람 1>> 스폰지밥 010-0000-0000
사람 2>> 뚱이 011-1111-1111
사람 3>> 징징이 012-2222-2222
모든 사람의 이름은 스폰지밥 뚱이 징징이
전화번호 검색합니다. 이름을 입력하세요>>스폰지밥
전화 번호는 010-0000-0000
```

빈칸 없이 입력

실습과제(4장)

10. 다음에서 Person은 사람을, Family는 가족을 추상화한 클래스로서 완성되지 않은 클래스이다.

```
class Person {
    string name;
public:
    Person (string name) { this->name = name; }
    string getName() { return name; }
};

class Family {
    Person *p; // Person 배열 포인터
    int size; // Person 배열의 크기. 가족 구성원 수
public:
    Family(string name, int size): // size 개수만큼 Person 배열 동적 생성
    void show(); // 모든 가족 구성원 출력
    ~Family();
};
```

실습과제(4장)

(앞 슬라이드에 이어) 다음 main()이 작동하도록 Person과 Family 클래스에 필요한 멤버들을 추가하고 코드를 완성하라.

```
int main() {
    Family *simpson= new Family("Simpson", 3); // 3명으로 구성된 Simpson 가족
    simpson->setName(0, "Mr. Simpson");
    simpson->setName(1, "Mrs. Simpson");
    simpson->setName(2, "Bart Simpson");
    simpson->show();
    delete Simpson;
}
```

```
Simpson 가족은 다음과 같이 3명 입니다.
Mr. Simpson      Mrs. Simpson      Bart Simpson
```

실습과제(4장)

11. 다음은 커피자판기로 작동하는 프로그램을 만들기 위해 필요한 두 클래스이다.

```
class CoffeeVendingMachine { // 커피자판기를 표현하는 클래스
    Container tong[3]; // tong[0]는 커피, tong[1]은 물, tong[2]는 설탕통을 나타냄
    void fill(); // 3개의 통을 모두 10으로 채움
    void selectEspresso(); // 에스프레소를 선택한 경우, 커피 1, 물 1 소모
    void selectAmericano(); // 아메리카노를 선택한 경우, 커피 1, 물 2 소모
    void selectSugarcoffee(); // 설탕커피를 선택한 경우, 커피 1, 물 2 소모, 설탕 1 소모
    void show(); // 현재 커피, 물, 설탕의 잔량 출력
public:
    void run(); // 커피 자판기 작동
};

class Container {  // 통 하나를 나타내는 클래스
    int size; // 현재 저장 량, 최대 저장량은 10
public:
    Container() { size = 10; }
    void fill(); // 최대량(10)으로 채우기
    void consume(); // 1 만큼 소모하기
    int getSize(); // 현재 크기 리턴
};
```

실습과제(4장)

(앞 슬라이드에 이어) 다음과 같이 실행되도록 main() 함수와 CoffeeVendingMachine, Container를 완성하라. 만일 커피, 물, 설탕 중 잔량이 하나라도 부족해 커피를 제공할 수 없는 경우'원료가 부족합니다.'를 출력하라.

```
***** 커피자판기를 작동합니다. *****
메뉴를 눌러주세요 (1:에스프레소, 2:아메리카노, 3:설탕커피, 4:잔량보기, 5:채우기)>> 4
커피 10, 물 10, 설탕 10
메뉴를 눌러주세요(1: 에스프레소, 2:아메리카노, 3:설탕커피, 4:진보기, 5:채우기)>> 1
에스프레소 드세요
메뉴를 눌러주세요(1:에스프레소, 2:아메리카노, 3:설탕커피, 4:잔량보기, 5: 채우기)>> 4
커피 9, 물 9, 설탕 10
메뉴를 눌러주세요 (1: 에스프레소, 2:아메리카노, 3:설탕커피, 4:진량보기, 5:채우기)>> 3
설탕커피 드세요
메뉴를 눌러주세요 (1:에스프레소, 2:아메리카노, 3:설탕커피, 4:잔량보기, 5: 채우기)>> 4
커피 8, 물 7, 설탕 9
메뉴를 눌러주세요(1:에스프레소, 2:아메리카노, 3: 설탕커피, 4:진량보기, 5:채우기)>> 5
커피 10, 물 10, 설탕 10
메뉴를 눌러주세요 (1:에스프레소, 2:아메리카노, 3:설탕커피, 4:진량보기, 5: 채우기)>>
```

실습과제(4장)

12. 다음은 이름과 반지름을 속성으로 가진 Circle 클래스와 이들을 배열로 관리하는 Circlemanager 클래스이다.

```
class Circle {
    int radius; // 원의 반지름 값
    string name; // 원의 이름
public:
    void setCircle(string name, int radius); // 이름과 반지름 설정
    double getArea();
    string getName();
};
```

```
class CircleManager {
    Circle *p; // Circle 배열에 대한 포인터
    int size; // 배열의 크기
public:
    CircleManager(int size); // size 크기의 배열을 동적 생성. 사용자로부터 입력 완료
    ~CircleManager ();
    void searchByName(); // 사용자로부터 원의 이름을 입력받아 면적 출력
    void searchByArea(); // 사용자로부터 면적을 입력받아 면적보다 큰 원의 이름 출력
};
```

실습과제(4장)

(앞 슬라이드에 이어) 키보드에서 원의 개수를 입력받고, 그 개수만큼 원의 이름과 반지름을 입력받고, 다음과 같이 실행되도록 main() 함수를 작성하라. Circle, CircleManager 클래스도 완성하라.

```
원의 개수 >> 4
원 1의 이름과 반지름 >> 빈대떡 10
원 2의 이름과 반지름 >> 도넛 2
원 3의 이름과 반지름 >> 초코파이 1
원 4의 이름과 반지름 >> 피자 15
검색하고자 하는 원의 이름 >> 도넛
도넛의 면적은 12.56
최소 면적을 정수로 입력하세요 >> 10
10보다 큰 원을 검색합니다.
빈대떡의 면적은 314, 도넛의 면적은 12.56, 피자의 면적은 706.5,
```

실습과제(4장)

13. 영문자로 구성된 텍스트에 대해 각 알파벳에 해당하는 문자가 몇 개인지 출력하는 히스토그램 클래스 Histogram을 만들어보자. 대문자는 모두 소문자로 변환하여 처리한다.

Histogram 클래스를 활용하는 사례와 실행 결과는 다음과 같다.

```
Histogram elvisHisto("Wise men say, only fools rush in But I can't help, ");
elvisHisto.put ("falling in love with you");
elvisHisto.putc('-');
elvisHisto.put ("Elvis Presley");
elvisHisto.print();
```

실습과제(4장)

Wise men say, only fools rush in But I can't help,
falling in love with you-Elvis Presley

총 알파벳 수 69

a (3) : ***
b (1) : *
c (1) : *
d (0) :
e (7) : *******
f (2) : **
g (1) : *
h (3) : ***
i (7) : *******
j (0) :
k (0) :
l (8) : ********
m (1) : *
n (6) : ******
o (5) : *****
p (2) : **
q (0) :
r (2) : **
s (6) : ******
t (3) : ***
u (3) : ***
v (2) : **
w (2) : **
x (9) : *********
y (4) : ****
z (0) :

실습과제(4장)

14. 갬블링 게임을 만들어보자. 두 사람이 게임을 진행하며, 선수의 이름을 초기에 입력받는다. 선수가 번갈아 자신의 차례에서 <Enter> 키를 치면 랜덤한 3개의 수가 생성되고 모두 동일한 수가 나오면 게임에서 이기게 된다. 숫자의 범위가 너무 크면 3개의 숫자가 일치할 가능성이 낮아 숫자의 범위를 0~2로 제한한다. 랜덤 정수 생성은 문제 3번의 힌트를 참고하라. 선수는 Player 클래스로 작성하고, 2명의 선수는 배열로 구성하라. 그리고 게임은 GamblingGame 클래스로 작성하라.

```
***** 갬블링 게임을 시작합니다. *****
첫번째 선수 이름>>수연이
두번째 선수 이름>>제갈이
수연이: <Enter>
                2    1    2    아쉽군요!
제갈이:<Enter>
                1    0    2    아쉽군요!
수연이:<Enter>
                2    2    1    아쉽군요!
제갈이:<Enter>
                2    0    2    아쉽군요!
수연이: <Enter>
                0    0    0    수연이님 승리!!
```