dongguk UNIVERSITY

심화프로그래밍

실습강의 12주차

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. :김동국19919876실습12.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**

일반화와 템플릿

- 템플릿(template)
  - 함수나 클래스 코드를 찍어내듯 생산할 수 있도록 일반화(generic) 시키는 도구

- 중복 함수의 일반화
  - 중복 함수들을 일반화시킨 특별한 함수를 템플릿 함수(template function) 혹은 제네릭 함수(generic function)라고 부른다.
  - Template 키워드를 사용하여 일반적인 함수를 템플릿 함수로 바꾸는 것을 일반화 시키는 과정이라고 한다.

```
template <class T> // 템플릿 선언과 제네릭 타입 T 선언
void myswap(T &a, T &b){
    T tmp;
    tmp = a;
    a = b;
    b = tmp;
}
```

  - template: 템플릿 선언하는 키워드
  - class: 제네릭 타입을 선언하는 키워드
  - T: 제네릭 타입 T, 기본 타입을 일반화 시킨 새로운 타입

템플릿으로부터의 구체화

- 구체화(specialization)
  - 중복 함수들을 템플릿화하는 과정의 역과정
- 구체화된 함수(specialization function)
  - 컴파일러가 함수의 호출문을 컴파일하여 구체화를 통해 제네릭 함수로부터 구체적인 함수의 소스 코드를 만들어 내는데, 이때 구체화를 통해 생성되는 함수

- 구체화하여 컴파일하는 과정

```
int a = 4, b = 5;
myswap(a, b); // 제네릭 타입 T에 int를 대입하여 구체화시킨 함수를 생성하여 호출
```

  - 컴파일러는 myswap(a, b); 호출문을 컴파일 할 때 myswap() 함수를 탐색함
  - 템플릿으로 선언된 myswap()함수 발견
  - 구체화: myswap(a, b); 의 함수 호출문에서 실인자 a,b 모두 int타입이므로 템플릿의 제네릭 타입 T에 int 를 대입시켜 구체화된 버전의 myswap(int &a, int &b)의 소스코드를 만들어 냄
  - 구체화된 함수의 소스 코드를 컴파일하고, 이 함수를 호출하도록 컴파일

템플릿

- 템플릿 역할
  - 템플릿 함수는 함수의 '틀'이다.
  - 제네릭 함수를 선언하고 컴파일 시점에 구체화시키기 위한 틀을 만드는 것.
  - 객체를 생성하지 않음

- 템플릿의 장단점
  - 소프트웨어의 생산성과 유연성을 높임(함수 작성 용이, 함수 코드 재사용)
  - 포팅에 약함(컴파일러에 따라 템플릿이 지원되지 않을 수도 있음)
  - 템플릿과 관련된 컴파일 오류 메시지가 빈약(디버깅 어려움)

제네릭 프로그래밍(generic programming)

- 템플릿을 이용해 제네릭 함수/클래스를 만들고 활용하여 프로그램을 작성하는 것

- 제네릭과 매크로의 차이점
  - 복잡한 함수나 클래스를 표현하는데 한계 존재
  - 타입 안전성(type-safe)이 확보되지 않아 실행 중에 부작용을 초래할 가능성이 높다
  - 템플릿은 제네릭 타입에 적용되는 실제 타입을 검사하여 구체화 과정을 거치기 때문에 타입 안정성이 확보된다.

제네릭 함수(Generic Function)

- 하나의 제네릭 타입을 가진 경우

```
template <class T>
T bigger(T a, T b) // 두 개의 매개 변수를 비교하여 큰 값을 리턴
```

```
#include <iostream>
using namespace std;
template <class T>
T bigger(T a, T b) { // 두 개의 매개 변수를 비교하여 큰 값을 리턴
    if(a > b) return a;
    else return b;
}
int main() {
    int a=20, b=50;
    char c='a', d='z';
    cout << "bigger(20, 50)의 결과는 " << bigger(a, b) << endl;
    cout << "bigger('a', 'z')의 결과는 " << bigger(c, d) << endl;
}
```

제네릭 함수(Generic Function)

- 두개 이상의 제네릭 타입을 가진 경우

```
#include <iostream>
using namespace std; // 두 개의 제네릭 타입 T1, T2를 가지는 copy()의 템플릿
template <class T1, class T2>
void mcopy(T1 src [], T2 dest [], int n) { // src[]의 n개 원소를 dest[]에 복사하는 함수
    for(int i=0; i<n; i++)
        dest[i] = (T2)src[i]; // T1 타입의 값을 T2 타입으로 변환한다.
}
int main() {
    int x[] = {1,2,3,4,5};
    double d[5];
    char c[5] = {'H', 'e', 'l', 'l', 'o'}, e[5];

    mcopy(x, d, 5); // int x[]의 원소 5개를 double d[]에 복사
    mcopy(c, e, 5); // char c[]의 원소 5개를 char e[]에 복사

    for(int i=0; i<5; i++) cout << d[i] << ' '; // d[] 출력
    cout << endl;
    for(int i=0; i<5; i++) cout << e[i] << ' '; // e[] 출력
    cout << endl;
}
```

연습문제(10장)

1. 일반화와 템플릿에 대해 잘못 설명한 것은?
   ① 템플릿은 C++에서 일반화를 위한 도구이다.
   ② 템플릿을 이용하여 함수와 클래스를 일반화할 수 있다.
   ③ 템플릿을 선언하기 위해 사용하는 키워드는 template이나 generic이다.
   ④ 제네릭 타입을 선언하기 위해 사용하는 키워드는 class이다.

2. 템플릿에 대해 잘못 말한 것은?
   ① 템플릿을 사용하면 소프트웨어 생산성과 유연성이 높아진다.
   ② 컴파일러에 따라 템플릿을 지원하지 않을 수 있기 때문에 포팅에 취약하다.
   ③ 템플릿을 사용하면 컴파일 오류 메시지가 풍부하여 디버깅에 많은 도움을 준다.
   ④ 제네릭 프로그래밍이라는 새로운 프로그래밍 패러다임을 가져왔다.

3. 다음에서 템플릿 선언을 잘못한 것은?
   ① `template <class T>`
   ② `template (class T)`
   ③ `template <typename T>`
   ④ `template <typename T1, typename T2>`

연습문제(10장)

4. 구체화의 과정은 누구에 의해 이루어지는가?

① 개발자 ② 컴파일러 ③ 로더 ④ 운영체제

5. 다음 두 함수를 일반화한 제네릭 함수를 작성하라.

```
bool equal(int a, int b) {
   if(a == b) return true;
   else return false;
}

bool equal(char a, char b) {
   if(a == b) return true;
   else return false;
}
```

연습문제(10장)

6. 다음 두 함수들을 일반화한 제네릭 함수를 작성하라.

```
void insert(int a, int b[], int index) {
   b[index] = a;
}

void insert(char a, char *b, int index) {
   *(b+index)= a;
}
```

7. 다음 제네릭 함수 선언에서 잘못된 부분을 바르게 고쳐라.

```
template <typename T> int max(T x, T y) {
   if(x > y) return x;
   else return y;
}
```

연습문제(10장)

8. 다음 제네릭 클래스의 선언에서 잘못된 부분을 바르게 고쳐라.

```
template <class TYPE>
TYPE equals(TYPE x, int y) {
   if(x == y) return true;
   else return false;
}
```

9. 다음 제네릭 함수가 있다.

```
template <class T> T avg(T *p, int n) {
   int k;
   T sum=0;
   for(k=0; k<n; k++) sum += p[k];
   return sum/n;
}
```

아래의 호출을 컴파일하여 생성되는 구체화된 버전의 avg() 함수의 소스 코드는 무엇인가?

(1)
```
int a[] = {1,2,3,4,5};
cout << avg(a, 5);
```

(2)
```
double d[] = {3.5, 6.7, 7.8};
cout << avg(d, 3);
```

연습문제(10장)

10. 다음 두 개의 함수가 있을 때, 질문에 답하여라.

```
template <class T> void show(T a) {
   cout << a;
}

void show(int a) {
   cout << "special " << a;
}
```

(1) 이 두 함수가 공존할 수 있는가?
(2) 만일 (1)의 답이 '예'라면, show(3.14);를 호출한 결과는?
(3) 만일 (1)의 답이 '예'라면, show(100);을 호출한 결과는?

11. 템플릿에 대한 설명 중 맞는 것은?

① 이 기능은 C++에만 있다.
② 컴파일러는 템플릿 함수나 클래스를 컴파일하여 일반화된 바이너리 코드를 생성한 후, 필요한 구체화를 시행한다.
③ 템플릿 함수와 동일한 이름의 함수가 중복되어 있을 때, 중복 함수가 우선적으로 바인딩된다.
④ 템플릿 함수를 선언할 때 디폴트 매개 변수를 선언할 수 없다.

연습문제(10장)

12. 템플릿 클래스 Container를 작성하고자 한다.

```
template <class T> class Container {
   _____ // T 타입의 포인터 p를 선언하라.
   _____ // 배열의 크기를 나타내는 변수 size를 선언하라.
public:
   Container(int n); // 멤버 변수 p에 n개의 동적 배열을 할당받는 생성자
   ~Container();
   void set(int index, T value) { p[index] = value; } // index 위치에 value 저장
   T get(int index); // index 위치의 값 리턴
};
```

(1) 빈칸을 적절하게 채워라.

(2) 생성자를 작성하라.

(3) 소멸자를 작성하라.

(4) get()을 작성하라.

(5) char 타입의 문자만 저장 가능한 Container 객체 c를 생성하는 선언문을 작성하라(c의 크기는 26).

(6) 문제 (5)에서 생성한 객체 c에 set() 함수를 이용하여 알파벳 'a'~'z'를 삽입하고, get() 함수를 이용하여 반대순으로 화면에 출력하는 main() 함수를 작성하라.

실습과제(10장)

1. 두 개의 배열을 비교하여 같으면 true를, 아니면 false를 리턴하는 제네릭 함수 equalArrays()를 작성하라. 또한 main() 함수를 작성하여 equalArrays ()를 호출하는 몇 가지 사례를 보여라. equalArrays()를 호출하는 코드 사례는 다음과 같다.

```
int x[] = {1, 10, 100, 5, 4};
int y[] = {1, 10, 100, 5, 4};
if(equalArrays(x, y, 5)) cout << "같다"; // 배열 x, y가 같으므로 "같다" 출력
else cout << "다르다";
```

2. 배열에서 원소를 검색하는 search() 함수를 템플릿으로 작성하라. search()의 첫 번째 매개 변수는 검색하고자 하는 원소 값이고, 두 번째 매개 변수는 배열이며, 세 번째 매개 변수는 배열의 개수이다. search() 함수가 검색에 성공하면 true를, 아니면 false를 리턴한다. search()의 호출 사례는 다음과 같다.

```
int x[] = {1, 10, 100, 5, 4};
if(search(100, x, 5)) cout << "100이 배열 x에 포함되어 있다"; // 이 cout 실행
else cout << "100이 배열 x에 포함되어 있지 않다";
```

실습과제(10장)

3. 다음 함수는 매개 변수로 주어진 int 배열 src에서 배열 minus에 들어있는 같은 정수를 모두 삭제한 새로운 int 배열을 동적으로 할당받아 리턴한다. retSize는 remove() 함수의 실행 결과를 리턴하는 배열의 크기를 전달받는다.

```
int* remove(int src[], int sizeSrc, int minus[], int sizeMinus, int& retSize);
```

템플릿을 이용하여 remove를 일반화하라.