객체지향설계
Fall 2018

임성수 교수

Week 11: 템플릿

수업 내용

1. 함수 템플릿
2. 클래스 템플릿

공지

향후 진행 계획

11주차 - 11/16 금	**템플릿**	**(30분 수업 + 창의SW축전 참관)**
12주차 - 11/19 월	프로젝트 중간 발표	발표 5~8분 + 질문/답변
12주차 - 11/23 금	휴강	(출장으로 인한 휴강)
13주차 - 11/26 월	실습	
13주차 - 11/30 금	디자인 패턴 소개	
14주차 - 12/03 월	실습	
14주차 - 12/07 금	주요 디자인 패턴	
추석보충 - 12/10 월	기말고사	
15주차 - 12/17 월	프로젝트 최종 발표	최종 보고서 제출
15주차 - 12/21 금	종강	최종 성적 확인

중간 발표 슬라이드 제출: 11/19 월 수업 시간 10분 전까지 sungsu@cnu.ac.kr

1. 함수 템플릿

일반화 프로그래밍 (Generic programming)

- 일반적인 코드를 작성하고 이 코드를 정수나 문자열 같은 다양한 타입의 객체에 대하여 재사용하는 프로그래밍 기법

템플릿 (Template)

- 템플릿: 물건을 만들 때 사용되는 틀이나 모형을 의미
- **함수 템플릿**: 함수를 찍어내기 위한 틀

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

get_max(1,3)으로 호출

T를 int로 치환

```
int get_max(int x, int y) {
  if (x > y)
    return x;
  else
    return y;
}
```

1. 함수 템플릿

[예제] 템플릿 함수

```
#include "pch.h"
#include <iostream>
using namespace std;

template <typename T>
T get_max(T x, T y) // 템플릿 함수
{
  if (x > y)
    return x;
  else
    return y;
}

int main() // 입력값에 맞게 치환 후 실행
{
  cout << get_max(1, 3) << endl;
  cout << get_max(2.8, 3.9) << endl;
  return 0;
}
```

실행 결과

```
3
3.9
```

1. 함수 템플릿

[예제] 템플릿 함수 특수화

```
#include "pch.h"
#include <iostream>
using namespace std;

template <typename T>
T sum(T a, T b)
{
  return a + b;
}

template<> // 아래와 같은 특수 케이스 고려
double sum(double a, double b)
{
  return a * b;
}

int main() // 입력이 double일 경우 특수화
{
  cout << sum(1, 2) << endl;
  cout << sum(3.3, 4.4) << endl;
  return 0;
}
```

실행 결과

```
3
14.52
```

1. 함수 템플릿

[예제] 템플릿 함수 오버로딩

```
#include "pch.h"
#include <iostream>
using namespace std;

template <typename T>
void swap_values(T x, T y) {
  T temp;
  temp = x;
  x = y;
  y = temp;
  cout << x << " " << y << endl;
}

void swap_values(char A, char B) {
  cout << "Overloading" << endl;
}

int main() {// 특수화 대신 오버로딩 가능
  swap_values(1, 2);
  swap_values('A', 'B');
  return 0;
}
```

실행 결과

```
2 1
Overloading
```

2. 클래스 템플릿

템플릿

- 템플릿: 물건을 만들 때 사용되는 틀이나 모형을 의미
- **클래스 템플릿**: 클래스를 찍어내기 위한 틀

```
template <typename T>
class Box {
  T data;
public:
  Box() {}
  void set(T value) {
    data = value;
  }
  T get() {
    return data;
  }
};
```

set(100)으로 호출

```
class Box {
  int data;
public:
  Box() {}
  void set(int value) {
    data = value;
  }
  int get() {
    return data;
  }
};
```

2. 클래스 템플릿

[예제] 클래스 템플릿

```
#include "pch.h"
#include <iostream>
using namespace std;

template <typename T>
class Box { // 템플릿 클래스
  T data;
public:
  Box() {}
  void set(T value) {
    data = value;
  }
  T get() {
    return data;
  }
};
```

```
int main() {
  Box<int> box1;
  box1.set(100);
  cout << box1.get() << endl;

  Box<double> box2;
  box2.set(99.9);
  cout << box2.get() << endl;

  return 0;
}
```

실행 결과

```
100
99.9
```

2. 클래스 템플릿

[예제] 클래스 템플릿

```
template <typename T>
class Box {
  T data;
public:
  Box() {}
  void set(T value) {
    data = value;
  }
  T get() {
    return data;
  }
};
```

```
template <typename T>
class Box { // 템플릿 클래스
T data;
public:
Box();
void set(T value);
T get();
};
```

```
template <typename T>
Box<T>::Box() { // 템플릿 클래스 함수
}
```

```
template <typename T>
void Box<T>::set(T value) {
data = value;
}
```

```
template <typename T>
T Box<T>::get() {
return data;
}
```

2. 클래스 템플릿

[예제] 클래스 템플릿

```
#include "pch.h"
#include <iostream>
using namespace std;

template <typename T1, typename T2>
class Box {
  T1 first_data;
  T2 second_data;
public:
  Box() {};
  T1 get_first() { return first_data; };
  T2 get_second() { return second_data; };
  void set_first(T1 value) { first_data = value; };
  void set_second(T2 value) { second_data = value; };
};

int main()
{
  Box<int, double> b;
  b.set_first(100);
  b.set_second(99.9);
  cout << "(" << b.get_first() << "," << b.get_second() << ")" << endl;
  return 0;
}
```

실행 결과

```
100
99.9
```

질문 및 답변