컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 52강 - C++의 템플릿

Fast campus

학습 목표

C++의 템플릿

1) C++의 템플릿 기법에 대해서 이해하고 이를 실제로 구현할 수 있습니다.

C++의 템플릿

일반화

C++은 일반화 프로그래밍(Generic Programming)이 가능한 언어입니다. C++에서는 템플릿(Template)을 이용해서 일반화 프로그래밍을 사용할 수 있습니다.

C++의 템플릿

템플릿

템플릿(Template)이란 매개변수의 타입에 따라서 함수 및 클래스를 손쉽게 사용할 수 있도록 해줍니다.

템플릿은 그 타입 자체가 매개변수에 의해서 다루어 집니다. 따라서 템플릿을 사용하면 타입마다 별도의 함수나 클래스를 만들지 않고, 다양한 타입에서 동작할 수 있는 단 하나의 객체를 정의할 수 있습니다. 결과적으로 소스코드의 재사용성을 극대화할 수 있는 객체 지향 프로그래밍 기법 중 하나입니다.

C++의 템플릿

함수 템플릿

```
#include <iostream>
#include <string>

using namespace std;

template <typename T>
void change(T& a, T& b) {
  T temp;
  temp = a;
  a = b;
  b = temp;
}

int main(void) {
  string a = "나동빈";
  string b = "홍길동";
  change(a, b);
  cout << a << ":" << b << endl;
  system("pause");
}
```

C++의 템플릿

함수 템플릿

함수 템플릿(Function Template)이 각각의 자료형에 대해서 처음으로 호출이 될 때, C++ 컴파일러는 해당 타입의 인스턴스를 생성하게 됩니다. 이후에 생성된 하나의 인스턴스는 해당 자료형에 대해서 특수화가 이루어집니다. 이러한 인스턴스는 해당 함수 템플릿에 해당 자료형이 사용될 때마다 호출됩니다.

C++의 템플릿

명시적 특수화

C++의 함수 템플릿은 특정한 타입에 대하여 명시적 특수화(Explicit Specialization) 기능을 제공합니다. 이러한 명시적 특수화를 이용하면 특정한 타입에 대해서 특수한 기능을 정의할 수 있습니다.

컴파일러는 호출된 함수에 대응하는 특수화된 정의를 발견한 이후에는 해당 정의만을 사용합니다.

C++의 템플릿

명시적 특수화

```
#include <iostream>
#include <string>

using namespace std;

template <typename T>
void change(T& a, T& b)
{
  T temp;
  temp = a;
  a = b;
  b = temp;
}

template <> void change<int>(int& a, int& b)
{
  cout << "정수형 데이터를 교체합니다.\n";
  int temp;
  temp = a;
  a = b;
  b = temp;
}
```

C++의 템플릿

클래스 템플릿

C++은 클래스를 일반화하기 위해서 클래스 템플릿(Class Template)을 활용할 수 있습니다. 클래스 템플릿을 사용하면 자료형에 따라서 다르게 동작하는 클래스 집합을 만들 수 있습니다.

C++의 템플릿

클래스 템플릿

```
#include <iostream>
#include <string>

using namespace std;

template <typename T>
class Data {
private:
  T data;
public:
  Data(T data) : data(data) { }
  void setData(T data) { this->data = data; }
  T getData() { return data; }
};

int main(void) {
  Data<int> data1(1);
  Data<string> data2("나동빈");
  cout << data1.getData() << ":" << data2.getData() << "\n";
  system("pause");
}
```

Fast campus

C++의 템플릿

클래스 템플릿: 디폴트 템플릿 인수

```
#include <iostream>
#include <string>

using namespace std;

template <typename T = int>
class Data {
private:
  T data;
public:
  Data(T data) : data(data) { }
  void setData(T data) { this->data = data; }
  T getData() { return data; }
};

int main(void) {
  Data<> data1(1);
  Data<string> data2("나동빈");
  cout << data1.getData() << ":" << data2.getData() << "\n";
  system("pause");
}
```

배운 내용 정리하기

C++의 템플릿

1) C++은 템플릿(Template) 문법을 활용해서 일반화(Generalization)을 적용할 수 있습니다.