Pervasive Elastic MetaLearning Laboratory
Department of Computer Engineering
Hongik University

DS 실습 2 - 템플릿

template

- C++ 프로그래밍 언어의 한 기능으로 함수나 클래스가 개별적으로 다시 작성하지 않고도 각기 다른 수많은 자료형에서 동작할 수 있게 한다. → 일반화 프로그래밍

```
int max(int a, int b) {
    return a > b ? a : b;
}

double max(double a, double b) {
    return a > b ? a : b;
}

float ...

long ....
```

```
template <typename Type>
Type max(Type a, Type b) {
    return a > b ? a : b;
}
```

템플릿 함수(1)

- 클래스와 함수의 재사용에 기여

(1) 템플릿 함수를 사용하지 않는 경우

- (예) 정수 배열을 위한 선택 정렬 함수 SelectionSort
  - 부동 소수점 수의 배열을 정리하려면?
    - 첫째 행에서 int *a를 float *a로 변경
    - 둘째 행에서 "정수"를 "부동 소수점 수"로 변경
    - 11번째 행에서 int를 float로 변경

(2) 템플릿 / 인수화타입(parameterized type)을 이용한 해결

- 소프트웨어 개발 시간과 저장 공간을 상당히 절약

템플릿 함수(2)

- 템플릿 함수 SelectionSort

```
template <class T>
void SelectionSort(T *a, const int n)
{// sort a[0] to a[n-1] into nonincreasing order.
    for ( int i = 0; i < n; i++) {
       int j = i;
       // find smallest integer in a[i] to a[n-1]
       for (int k = i + 1; k < n; k++)
         if (a[k] < a[j]) j = k;
       swap(a[i], a[j]);
    }
}
```

- 템플릿 함수 SelectionSort 이용 예

```
float farray[100];
int intarray[250];
...
// 이 시점에서 배열들이 초기화된다고 가정한다
SelectionSort(farray, 100);
SelectionSort(intarray, 250);
```

템플릿 함수(3)

- 템플릿 함수 SelectionSort 사용시 주의사항
  - <, = 및 복사 생성자
    - int와 float에 대해서는 자동적으로 정의
    - 사용자 정의 데이터 타입에 대해서는 별도로 정의하여야 함
  - (예) SelectionSort를 이용해 Rectangle 배열을 넓이 순으로 정렬
    - operator < 정의 필요
    - 지정 연산자, 복사 생성자는 컴파일러의 묵시적 구현으로 충분

- 1-dim array의 크기 변경하는 템플릿 함수 ChangeSize1D

```
template <class T>
void ChangeSize1D(T *& a, const int oldSize, const int
newSize)
{
    if (newSize < 0) throw “New length must be >= 0”;
    T* temp = new T[newSize];    // new array
    int number = min(oldSize, newSize);
    copy(a, a + number, temp);
    delete [] a;
    a = temp;
}
```

template

- 템플릿 특수화

특정 타입이 들어왔을 때 다른 방식으로 처리하기 위한 방법

```cpp
#include <iostream>
using namespace std;
template <typename T>
T add(T x, T y) {
    return x + y;
}

template<>
char* add(char* s1, char* s2) {
    char* str = new char[100];
    strcpy(str, s1);
    strcat(str, s2);
    return str;
}

int main()
{
    char num1[] = "10", num2[] = "20";
    cout << add(num1, num2) << endl;

    return 0;
}
```

실습 1

1. hw2a.cpp 파일을 만든다.
2. 코드를 다음과 같이 작성하고 실행한다.
3. Template을 사용해 3개의 함수를 하나로 줄인다.
4. 실행결과를 확인한다.

```
[B611107@localhost ds_hw1]$ make hw1a
g++     -c -o hw1a.o hw1a.cpp
g++ -o hw1a hw1a.o
[B611107@localhost ds_hw1]$ hw1a
3
7.7
19.31
[B611107@localhost ds_hw1]$
```

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


int add(int x, int y){
        return x + y;
}

double add(double x, double y){
        return x + y;
}

float add(float x, float y){
        return x + y;
}


int main() {
        int intX = 1, intY = 2;
        double dbX = 3.3, dbY = 4.4;
        float fX= 09.24, fY = 10.07;

        cout << add(intX, intY) << endl;
        cout << add(dbX, dbY) << endl;
        cout << add(fX, fY) << endl;


        return 0;
}
```

실습 2

- 프로그램이 올바르게 동작하도록 다음의 코드를 완성하라(hw2b.cpp)
- 조건: Class 생성은 한 번만 가능! 함수는 선언과 정의를 분리 (함수 정의는 class 외부에서)
- 실행 결과

```
[B611107@localhost ds_hw1]$ make hw1b
g++     -c -o hw1b.o hw1b.cpp
g++ -o hw1b hw1b.o
[B611107@localhost ds_hw1]$ hw1b
(3, 5)
(3.3, 5.5)
[B611107@localhost ds_hw1]$
```

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


// Class ...

int main() {
        CPoint<int> P1(1, 2);
        CPoint<double> P2(1.1, 2.2);

        P1.move(2, 3);
        P2.move(2.2, 3.3);
        P1.print();
        P2.print();

        return 0;
}
```

실습 3

- 프로그램이 올바르게 동작하도록 다음의 코드를 완성하라(hw2c.cpp)
- 조건 : 실습 2와 동일
- 실행 결과

```
[B611107@localhost ds_hw1]$ make hw1c
g++     -c -o hw1c.o hw1c.cpp
g++ -o hw1c hw1c.o
[B611107@localhost ds_hw1]$ hw1c
(9, 24)
(10.07, 22.59)
[B611107@localhost ds_hw1]$
```

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


// Class ...

int main() {
        CPoint<int> P1(1, 2);
        CPoint<double> P2(1.1, 2.2);

        P1.move(8, 13);
        P2.move(8.97, 20.39);

        cout << P1 << P2;

        return 0;
}
```

실습 4

- 프로그램이 올바르게 동작하도록 다음의 코드를 완성하라(hw2d.cpp)
- 조건 : 실습 2와 동일
- 실행 결과

```
(4, 6)
(4.4, 6.6)
```

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


// Class ...

int main() {
        CPoint<int> P1(1, 2);
        CPoint<int> P2(3, 4);

        CPoint<double> P3(1.1, 2.2);
        CPoint<double> P4(3.3, 4.4);

        P1 = P1 + P2;
        P3 = P3 + P4;

        cout << P1 << P2;

        return 0;
}
```

질문

- pemds81718@gmail.com
- 간단한 구글링으로 알 수 있는 내용은 답변하지 않습니다.

- C++ 참고자료 : https://modoocode.com/219