2020.09.21 - 4주차

객체지향설계

Pointer

Content

1. 지난주 과제 풀이
2. C++ 문법
    1. cin
    2. pointer
    3. const
3. 과제
4. QnA

과제 01 풀이

- "실습 1"의 코드가 OOP의 추상화 및 캡슐화 개념을 만족하도록 바꾸어 보세요.
- 단, SetValue 클래스의 기존 구문은 수정하지 않고 새로운 멤버 함수만 추가하여 수정할 것.

  - ✓ 추상화 : 클래스 내부의 정보를 외부로부터 숨김
  - ✓ 캡슐화: 클래스 멤버 변수와 멤버 함수와의 연결 (binding) --〉 내부 변수 접근을 위한 인터페이스 제공

과제 02 풀이

- 다음 프로그램의 확장자를 .c 와 .cpp의 두 가지 경우로 실행(컴파일)해 보세요.
- 각 경우에 에러가 나거나 나지 않는 이유에 대해 생각해 보세요.

```c
#include <stdio.h>

void print(int var){
	printf("Integer number: %d ₩n", var);
}

void print(float var){
	printf("Float number: %f ₩n", var);
}

void print(int var1, float var2){
	printf("Integer number: %d ₩n", var1);
	printf(" and float number: %f", var2);
}
```

```c
int main(){
	int a = 7;
	float b = 9;

	print(a);
	print(b);
	print(a, b);

	return 0;
}
```

- c와 달리 c++ 은 함수 오버로딩을 지원하기 때문에 에러가 나지 않는다. 문제의 프로그램은 아래와 같이 c++ 형식으로도 바꿀 수 있다.

C++ 문법

- cin 객체
  - ✓ 데이터를 입력받는 데 사용되며 미리 정의된 입력 스트림을 나타내는 객체
  - ✓ std::cin 〉〉 저장할변수;

```
int main() {
std::string s;
std::cout << "문자를 입력하세요(100자 이내)." << std::endl;
std::cin >> s;
std::cout << "입력된 문자는 " << s << "입니다." << std::endl;

system("pause"); // keep terminal open
return 0;
}
```

C++ 문법

- Call by value
  - ✓ 값을 복사해 전달

```
#include <iostream>
void increment(int x){
    ++x;
}
int main(){
    int x = 55;
    std::cout << " Before increment: " << x << std::endl;
    increment(x);
    std::cout << " After increment: " << x << std::endl;

    system("pause");
    return 0;
}
```

C++ 문법

- 주소값
  - ✓ 데이터의 주소값이란 해당 데이터가 저장된 메모리의 시작 주소를 의미
  - ✓ C++에서는 이러한 주소값을 1바이트 크기의 메모리 공간으로 나누어 사용

- 포인터
  - ✓ C++에서 포인터(pointer)란 메모리의 주소값을 저장하는 변수이며, 포인터 변수라고도 함
  - ✓ char형 변수가 문자를 저장하고, int형 변수가 정수를 저장하는 것처럼 포인터는 주소값을 저장하는 데 사용

C++ 문법

➢ 포인터 연산자

✓ 주소연산자

- 주소 연산자는 변수의 이름 앞에 사용하여, 해당 변수의 주소값을 반환
- '&'기호는 앰퍼샌드(ampersand)라고 읽으며, 번지 연산자라고도 불림

✓ 참조연산자

- 참조 연산자는 포인터의 이름이나 주소 앞에 사용하여, 포인터에 저장된 주소에 저장되어 있는 값을 반환
- '*'기호는 역참조 연산자로 에스크리터(asterisk operator)라고도 불림

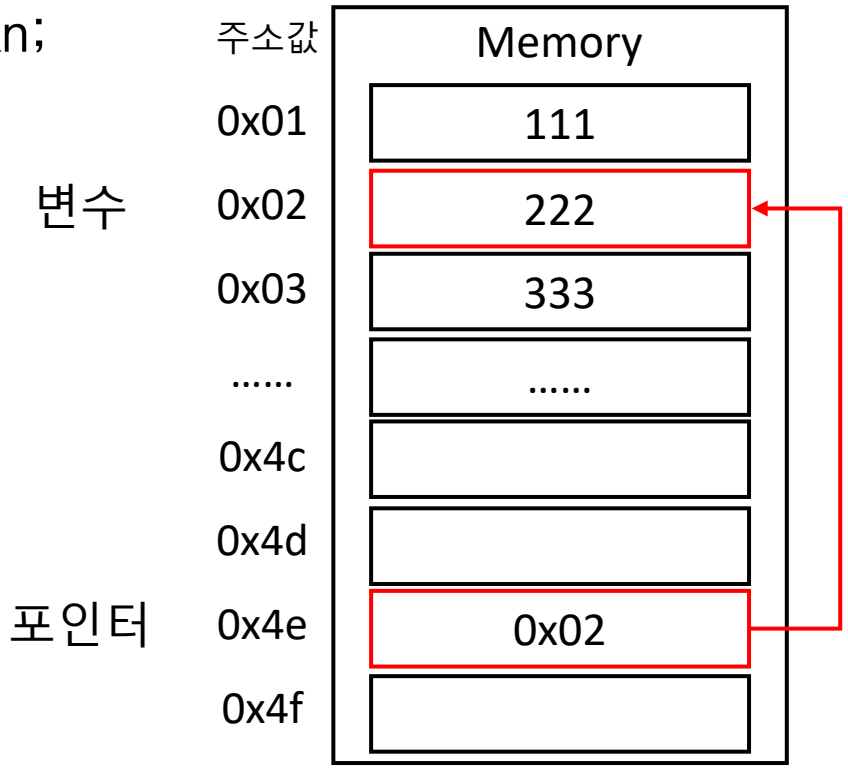
➢ 포인터 선언

✓ 타입 * 포인터 이름

C++ 문법

- 주소값, 포인터

```
int n = 222;
int *ptr = &n;
```



C++ 문법

- Call by reference
  - ✓ "포인터"를 활용하여 increment 함수의 인풋값이 +1 증가하도록 수정

```
#include <iostream>
void increment(int *x){
    ++ *x;
}
int main(){
    int x = 55;
    std::cout << " Before increment: " << x << std::endl;
    increment(&x);
    std::cout << " After increment: " << x << std::endl;

    system("pause");
    return 0;
}
```

C++ 문법

- C++ reference
  - ✓ "c++의 레퍼런스 (&)"를 활용하여 increment 함수의 인풋값이 +1 증가하여 프린트되도록 수정

```
#include <iostream>
void increment(int &x){
    ++x;
}
int main(){
    int x = 55;
    std::cout << " Before increment: " << x << std::endl;
    increment(x);
    std::cout << " After increment: " << x << std::endl;

    system("pause");
    return 0;
}
```

C++ 문법

- C++ const reference
  - ✓ const : 값을 상수로 선언할 수 있도록 도와주는 키워드
  - ✓ const 값에 대한 포인터를 선언하는 것처럼 const 값에 대한 참조 선언 가능

- x의 타입(래퍼런스)를 유지하며 아래 코드의 컴파일 에러를 해결하시오

```
#include <iostream>

int main(){
    int& x = 5;
    std::cout << x << std::endl;
}
```

과제 01

- "실습 1"의 코드를 C 코드로 변경해 보자
- 100자 이내로 입력된다고 가정하고 별도의 예외 처리는 하지 않아도 됨

homework01.cpp

```
int main() {
std::string s;
std::cout << "문자를 입력하세요(100자 이내)." << std::endl;
std::cin >> s;
std::cout << "입력된 문자는 " << s << "입니다." << std::endl;

system("pause"); // keep terminal open
return 0;
}
```

과제 02

- 아래 코드의 swap 함수는 인풋 파라미터를 서로 바꾸는 역할을 한다.
- 정상적으로 동작하도록 프로그램을 수정하세요. (힌트: 레퍼런스 활용)

homework02.cpp

```
#include<iostream>
void swap(int first, int second){
    int temp = first;
    first = second;
    second = temp;
}
int main(){
    int a = 2, b = 3;
    swap(a, b);
    std::cout << a << " " << b << std::endl;
    return 0;
}
```

과제 제출

- 제출
  - ✓ 09월 28일 24시 까지 사이버캠퍼스에 보고서 및 깃헙 링크 업로드
  - ✓ 지각 제출시 만점의 50%
  - ✓ Copy 적발시 0점

- 보고서 작성방법
  - ✓ 표지 : 제출일, 학번, 이름, 주차(week04)
  - ✓ 과제 수행 과정, 실행결과 캡쳐 후 결과분석
  - ✓ 과제를 수행하며 어려웠던 점 또는 새로 알게 된 부분을 고찰한 내용
  - ✓ 과제의 해답 코드를 깃허브에 올리고 링크를 보고서에 기입
  - ✓ 파일형식 : PDF

QnA

사이버캠퍼스 질문게시판 또는
ljy8972@naver.comfh 이나
ymc12377@naver.com 로 메일