함수 포인터

남동강

함수 포인터(function pointer)란?

- 함수를 가리키는 변수
- 함수의 주소를 저장하는 변수

- Callback
  - 특정한 사건 또는 메시지가 발발했을 때 호출되도록 지정하여 사용

- 코드의 유연성을 높이는 반면 가독성은 떨어진다

함수 포인터

```
void func(int value)
{
  cout << "func : " << value << endl;
}
  void func_print(void(*fp)(int), int value)
{
  fp(value);
}
int main()
{
  func_print(func, 1);
}
```

...

```
int main()
{
  void(*funcp)(int) = func;
  func_print(funcp, 1);
}
```

함수 포인터

```
typedef void(*FP)(int);
void func(int value)
{
  cout << "func : " << value << endl;
}
void func_print(FP func, int value)
{
  func(value);
}
int main()
{
  FP fp = func;
  func_print(fp, 1);
}
```

```
using FP = void(*)(int);
...
```

Callback 함수의 사용 예

```
//server
using CALLBACK = void(*)(int);
CALLBACK cb_func = NULL;
void RegistCallback(CALLBACK cb)
{
  cb_func = cb;
}
void StartCallback()
{
  if (NULL == cb_func)
  {
    cout << "등록된 Callback 함수가 없습니다." << endl;
    return;
  }
  cout << "Callback 호출!" << endl;
  cb_func(0);
}

//client
void UserCallback(int param)
{
  cout << "Parameter : " << param << endl;
}

//client
RegistCallback(UserCallback); //callback 등록.

//server
StartCallback(); //event 발생!! => Client의 UserCallback 함수가 호출된다!!
```

연습문제

- 원하는 형식의 함수 포인터를 만든다
- callback 함수를 등록하고 호출하는 class를 만든다
- main에 무한 루프를 만들어 0값이 들어오면 callback을 호출하고 루프를 빠져나오고 아니라면 해당 값을 출력하게 한다