포인터

(Pointer)

들어가기 전에

- 오늘의 소스코드를 받아봅시다.
  - 중요한 코드는 꼭 다른 곳으로 복사해두세요.
  - phycom2015-2 디렉토리에서,
    - git stash
    - git pull
  - 디렉토리가 사라졌다면,
    - git clone https://github.com/geonmo/phycom2015-2.git

지금까지의 내용

- 변수란 무엇인가?
  - 변수와 상수
- 제어문이란?
  - 조건문과 반복문
- 함수란?
  - 한 로직을 처리하기 위한 프로그램 코드의 묶음
- 자료구조란?
  - 데이터를 저장/관리하는 가장 효율적인 방법들

포인터란?

- 포인터는 무엇인가를 가리킬 수 있는 변수
  - 가리키기(Point) 때문에 "포인터"라고 부름
  - 값 그 자체가 아니라 값이 있는 "위치"를 저장
- 왜 쓰는가?
  - 실제 저장된 데이터가 너무 클 때
    - 예) 그림파일
  - 함수에서 값을 바꾸고 싶을 때 (call by reference)

포인터란?

- 포인터의 사용법
  - 선언
    - 가리키게 될 변수의 데이터형+"*"
      - 예) int형 : int* , double형 : double*
        std::string형 : std::string*
  - 정의
    - 이미 존재하는 변수로 정의 : &(Address연산자)
      - int* pa = &a;
    - 특정한 위치를 잡을 수 없을 경우
      - int* pa = NULL;
        - » NULL == (void)0
        - » c++11 이상부터는 std::nullptr 사용!

포인터란?

- 포인터의 사용법
  - 포인터 변수가 가리키고 있는 값은 포인터 변수 앞에 *를 붙여 확인 가능
    - 예) std::cout<<*pa<<std::endl;
  - 포인터 변수의 배열일 경우
    - 다음 변수로 이동 : pa+1 or pa++
    - 이전 변수로 이동 : pa-1 or pa—
    - 지정된 변수형 크기에 따라 이동이 됨
      - 즉, int는 4바이트 double은 8바이트씩 점프!

포인터란?

- 포인터 변수 하나로 모든 데이터의 배열을 전달 가능
  - 그림, 파일 등 대규모 정보를 전달할 때 매우 유리

5_pointer_for_small.cpp

```
#include<iostream>

using namespace std;
int main()
{
  int a[5]      = {32,45,21,11,62};
  int* pa = a;
  cout<<"Integer : "<<a[1]<<"₩t₩t size
of Integer variable :
"<<sizeof(a)<<std::endl;
  cout<<endl;
  cout<<"Interger pointer :
"<<*(pa+1)<<"₩t₩t size of Integer
pointer : "<<sizeof(pa)<<endl;
  return 0;
}
```

포인터란?

- Call by reference
  - 함수에서 2개 이상의 리턴 값을 필요로 할 때
    - 예) 두 변수의 값 교환

8_swap.cpp

```
#include<iostream>

int swap( int a, int b ) {
  std::cout<<"In swap ) a : "<<a<<" b : "<<b<<std::endl;
  int tmp = a;
  a = b;
  b = tmp;
  std::cout<<"In swap ) a : "<<a<<" b : "<<b<<std::endl;
  return 0 ;   // Normally terminated.
}

int pointer_swap( int* pa, int* pb) {
  std::cout<<"In pointer_swap ) a : "<<*pa<<" b : "<<*pb<<std::endl;
  int tmp = *pa;
  *pa = *pb;
  *pb = tmp;
  std::cout<<"In pointer_swap ) a : "<<*pa<<" b : "<<*pb<<std::endl;
  return 0 ;
}


int main()
{
  int a=32, b= 48;


  std::cout<<"In Main ) a : "<<a<<" b : "<<b<<std::endl;
  swap(a,b);
  std::cout<<"In Main ) a : "<<a<<" b : "<<b<<std::endl;


  std::cout<<std::endl;


  std::cout<<"In Main ) a : "<<a<<" b : "<<b<<std::endl;
  pointer_swap(&a,&b);
  std::cout<<"In Main ) a : "<<a<<" b : "<<b<<std::endl;

  return 0;
}
```

다차원 포인터

- 포인터 변수를 가리키고 있는 포인터
  - int a= 34;
  - int* pa = &a;
  - int** ppa = &pa;
- 해석하기가 어렵다면?
  - 2차원(**)부터 마지막 하나를 배열로 해석
    - int**는 int*의 배열과 거의 비슷
    - 비슷한 용도로 쓴다면 vector<int*>로 하는 것이 보다 명확하게 코드를 작성하는 방법

함수 포인터

- 함수 또한 포인팅 할 수 있다!
  - 함수의 이름은 "포인터"
  - 예) main, add, substract, swap 등

10_funtor.cpp

```
#include<iostream>

int add(int a, int b){
  std::cout<<"a+b = "<<a+b<<std::endl;
  return 0 ;
}

int substract( int a, int b) {
  std::cout<<"a-b = "<<a-b<<std::endl;
  return 0 ;
}


int main()
{
  int (*absFunction)(int , int );  // 함수 포인터의 선언!

  int a,b;
  char c;

  std::cin>>a>>c>>b;  // 14+12
  if ( c=='+') absFunction = add;
  else if ( c=='-') absFunction = substract;
  else {
    std::cerr<<"You did not input correctly. Terminate this
program"<<std::endl;
    return -1;
  }
  absFunction(a,b);
  return 0;
}
```