윈도우 프로그래밍

7. 객체 활용(1)

2018. 4.27.

심미나 교수

목 차

I. 객체 배열과 this포인터

객체 배열과 this포인터

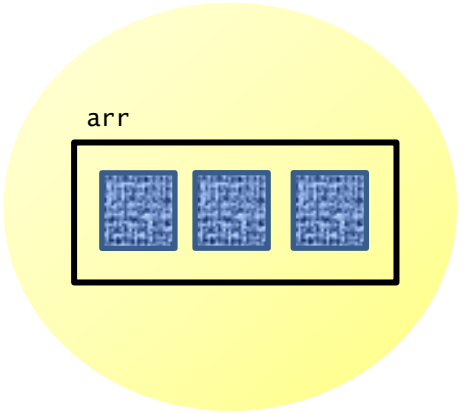
객체 배열과 객체 포인터 배열

- **객체 배열**
  - 객체로 이루어진 배열
  - 배열 생성시, 객체가 함께 생성
    - 호출되는 생성자는 void 생성자

```
Person arr[3];
Person *parr = new Person[3];
```

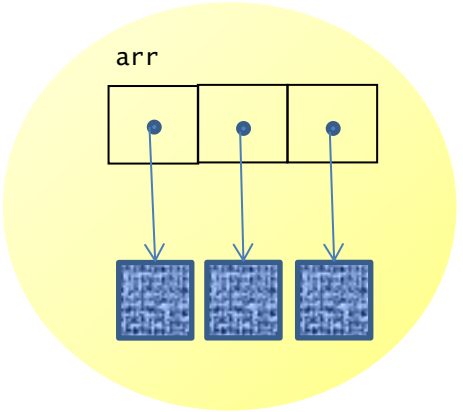
ⓘ Person객체 3개 묶인 배열

ⓘ Delete []parr; 로 삭제
ⓘ Delete parr[3]; (x)



- **객체 포인터 배열**
  - 객체를 저장할 수 있는 포인터 변수로 이루어진 배열
  - 별도의 객체 생성 과정 필요

```
Person *arr[3];
arr[0]=new Person(name, age);
arr[1]=new Person(name, age);
arr[2]=new Person(name, age);
```



※객체배열을 선언할지, 객체포인터배열을 선언할지 먼저 결정해야 함!

객체 배열과 this포인터

this 포인터의 이해

- this 포인터 - 사용된 객체 자신의 주소값을 갖는 포인터

```
int main(void)
{
    SoSimple sim1(100);
    SoSimple * ptr1=sim1.GetThisPointer();
    cout<<ptr1<<", ";
    ptr1->ShowSimpleData();
    SoSimple sim2(200);
    SoSimple * ptr2=sim2.GetThisPointer();
    cout<<ptr2<<", ";
    ptr2->ShowSimpleData();
    return 0;
}
```

1) 객체 생성

2) 객체 주소값 반환

3) 반환된 주소값 출력

4) 객체(주소)의 데이터 출력

실행 결과

```
Num=100, address=0012FF60
0012FF60, 100
Num=200, address=0012FF48
0012FF48, 100
```

```
class SoSimple
{
private:
    int num;
public:
    SoSimple(int n) : num(n)
    {
        cout<<"num="<<num<<", ";
        cout<<"address="<<this<<endl;
    }
    void ShowSimpleData()
    {
        cout<<num<<endl;
    }
    SoSimple * GetThisPointer()
    {
        return this;
    }
};
```

① 반환형(포인터)

① sim 자기자신

객체 배열과 this포인터

this 포인터의 활용

- 객체의 주소 값(this)으로 멤버변수에 접근 가능

```
class TwoNumber
{
private:
    int num1;
    int num2;
public:
    TwoNumber(int num1, int num2)
    {
        this->num1=num1;
        this->num2=num2;
    }
```

ⓘ 멤버변수

ⓘ 매개변수

ⓘ 멤버변수 접근 시 this 사용
this->num1 : 멤버변수 num1
즉, 객체주소(this)는 매개변수 접근에 사용 못하고, 멤버변수 접근 시에만 사용

```
TwoNumber(int num1, int num2)
    : num1(num1), num2(num2)
{
    // empty
}
```

ⓘ 멤버변수

ⓘ 이니셜라이저는 this-> 표현 사용 불가

객체 배열과 this포인터

Self-reference의 반환

- Self-reference - **객체 자신을 참조하는 참조명**

```
int main(void)
{
    SelfRef obj(3);
    SelfRef &ref=obj.Adder(2);
    obj.ShowTwoNumber();
    ref.ShowTwoNumber();
    ref.Adder(1).ShowTwoNumber().Adder(2).ShowTwoNumber();
    return 0;
}
```

ⓘ 객체자신을 반환

```
class SelfRef
{
private:
    int num;
public:
    SelfRef(int n) : num(n)
    {
        cout<<"객체생성"<<endl;
    }
    SelfRef& Adder(int n)
    {
        num+=n;
        return *this;
    }
    SelfRef& ShowTwoNumber()
    {
        cout<<num<<endl;
        return *this;
    }
};
```

ⓘ 단, 참조형으로 반환

ⓘ 객체자신을 반환

실행 결과

```
객체생성
5
5
6
8
```

객체 배열과 this포인터

Self-reference의 반환

- Self-reference - 객체 자신을 참조하는 참조명

```
int main(void)
{
    SelfRef obj(3);
    SelfRef &ref=obj.Adder(2);
    obj.ShowTwoNumber();
    ref.ShowTwoNumber();
    ref.Adder(1).ShowTwoNumber().Adder(2).ShowTwoNumber();
    return 0;
}
```

① 객체자신을 반환

```
class SelfRef
{
private:
    int num;
public:
    SelfRef(int n) : num(n)
    {
        cout<<"객체생성"<<endl;
    }
    SelfRef& Adder(int n)
    {
        num+=n;
        return *this;
    }
    SelfRef& ShowTwoNumber()
    {
        cout<<num<<endl;
        return *this;
    }
};
```

① 단, 반환형(참조자)

```
(ref.Adder(1)).ShowTwoNumber().Adder(2).ShowTwoNumber();
```

(1) ref참조자 반환

(2) (ref참조자).ShowTwoNumber()

(3) (ref참조자).Adder(2)

(4) (ref참조자).ShowTwoNumber()

IV. 실습

감사합니다

mnshim@sungkyul.ac.kr