



C++ 스터디 2주차

과제

```
1 #include<iostream>
2 using namespace std;
3
4 struct Point
5 {
6     int x, y;
7     int number;
8 };
9
10 int main(void)
11 {
12     int n;
13     Point* arr;
14
15     cin >> n;
16     arr = new Point[n];
17
18     for(int i =0;i<n;i++)
19     {
20         arr[i].number = i;
21         cin >> arr[i].x >> arr[i].y;
22     }
23
24     for(int i = 0; i < n - 1;i++)
25     {
26         int min = i;
27         for(int j = i + 1;j < n;j++)
28         {
29             if(arr[min].x + arr[min].y > arr[j].x + arr[j].y)
30                 min = j;
31         }
32
33         Point tmp = arr[i];
34         arr[i] = arr[min];
35         arr[min] = tmp;
36     }
37
38     for(int i =0;i<n;i++)
39     {
40         cout << "Point" << arr[i].number << " : " << arr[i].x << ", " << arr[i].y << endl;
41     }
42 }
```

과제

```
1 #include<iostream>
2 using namespace std;
3
4 struct Node
5 {
6     int data;
7     Node* next;
8 };
9
10 Node* first = NULL;
11 Node* last = NULL;
12
13 int main(void)
14 {
15     int menu;
16     int data;
17     Node* tmp;
18
19     while (true)
20     {
21         cout << "메뉴 선택 (1 : 입력, 2 : 출력, 3 : 종료) : ";
22         cin >> menu;
23
24         switch(menu)
```

```
24         switch(menu)
25         {
26             case 1:
27                 cout << "값 입력 : ";
28                 cin >> data;
29
30                 if(first == NULL)
31                 {
32                     first = last = new Node;
33                     first->data = data;
34                     first->next = NULL;
35                 }
36                 else
37                 {
38                     last = last->next = new Node;
39                     last->data = data;
40                     last->next = NULL;
41                 }
42                 break;
43             case 2:
44                 tmp = first;
45
46                 while(tmp)
```

과제

```
43         case 2:
44             tmp = first;
45
46             while(tmp)
47             {
48                 cout << tmp->data << ' ';
49                 tmp = tmp->next;
50             }
51             cout << endl;
52             break;
53         case 3:
54             return 0;
55     }
56 }
57 }
```

참조

- 기존 변수의 또 다른 이름(홀로 존재 불가)
- 선언과 동시에 초기화 필요
- & 사용

참조

```
1 #include<iostream>
2 using namespace std;
3
4 int main(void)
5 {
6     int var = 2;
7     int& refVar = var;
8
9     cout << "var : " << var << ", ref : " << refVar << endl;
10
11     refVar = 5;
12
13     cout << "var : " << var << ", ref : " << refVar << endl;
14
15     return 0;
16 }
```

@localhost study]\$ a.out

var : 2, ref : 2

var : 5, ref : 5

참조

- 값이나 수식은 참조 불가능

- `int& a = 3;` (x)

- `int& c = a + b;` (x)

참조

- swap 함수의 변경

```
void swap(int* a, int* b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

```
void swap(int& a, int& b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

참조 반환

- 반환 값이 그 변수 자체
- 지역 변수는 참조 반환 불가능
- 전역 변수, 정적 변수, 참조 매개 변수 반환 가능

참조 반환

```
1 #include<iostream>
2 using namespace std;
3 int& min(int&, int&);
4
5 int main(void)
6 {
7     int a = 2, b = 3;
8
9     min(a, b) = 5;
10
11     cout << "a : " << a << ", b : " << b << endl;
12
13     return 0;
14 }
15
16 int& min(int& x, int& y)
17 {
18     return (x < y) ? x : y;
19 }
```

실습

- N을 입력 받아서 길이가 N인 int형 배열 동적 생성.
- i번째 원소를 반환하는 함수 GetEl(int* arr, int i)를 작성하고, 이를 이용해 i번째 원소를 i로 초기화
- 출력해서 확인하기

```
[ @localhost study]$ a.out
N 입력 : 6
1번째 원소 : 1
2번째 원소 : 2
3번째 원소 : 3
4번째 원소 : 4
5번째 원소 : 5
6번째 원소 : 6
```

실습

```
1 #include<iostream>
2 using namespace std;
3 int& GetEl(int*, int);
4
5 int main(void)
6 {
7     int N;
8     int* arr;
9
10    cout << "N 입력 : ";
11    cin >> N;
12
13    arr = new int[N];
14
15    for(int i = 0; i < N; i++)
16        GetEl(arr, i) = i + 1;
17
18    for(int i = 0; i < N ; i++)
19        cout << i + 1 << "번 째 원 소 : " << GetEl(arr, i) << endl;
20
21    return 0;
22 }
23
24 int& GetEl(int* arr, int i)
25 {
26     return arr[i];
27 }
```

객체지향 프로그래밍



- 데이터 추상화
- 상속
- 다형성
- 일반화 프로그래밍

- 추상 데이터 형 : 속성 + 메서드
- 자료형 - 변수
- 클래스 - 객체

```
class Person
{
private:
    char name[10];
    int height;
    int weight;
public:
    void Set(char* n, int l, int h, int w)
    {
        for(int i =0;i<l;i++)
            name[i] = n[i];

        height = h;
        weight = w;
    }

    void Print()
    {
        cout << name << " : height - " << height << ", weight - " << weight << endl;
    }
};
```

```
int main(void)
{
    Person Kim;

    Kim.Set("Kim", 3, 180, 65);

    Kim.Print();


    return 0;
}
```

```
[redacted@localhost study]$ a.out
Kim : height - 180, weight - 65
```

접근 지정자

- **private** : 외부에서 접근 불가(클래스 내부에서만 사용)
- **public** : 외부에서 접근 가능
- 주로 멤버 변수는 **private**, 멤버 함수는 **public**
- **protected**는 나중에

생성자



- 객체가 생성되면 반드시 생성자가 호출됨
- 생성자 이름은 클래스 이름과 같음
- 반환형이 없고, 어느 것도 반환하지 않음
- 매개 변수 자유롭게 줄 수 있음 -> 오버로딩 가능
- default parameter 사용 가능

생성자

```
class Person
{
private:
    char name[10];
    int height;
    int weight;
public:
    Person(const char* n, int l, int h, int w)
    {
        for(int i = 0; i < l; i++)
            name[i] = n[i];
        height = h;
        weight = w;
    }

    void Print()
    {
        cout << name << " : height - " << height << ", weight - " << weight << endl;
    }
};

int main(void)
{
    Person Kim("Kim", 3, 180, 65);
    Person Lee = Person("Lee", 3, 175, 60);

    Kim.Print();
    Lee.Print();

    return 0;
}
```

생성자 오버로딩

```
class Point
{
    int x, y;
public:
    Point()
    {
        x = y = 0;
    }

    Point(int a, int b)
    {
        x = a;
        y = b;
    }

    void Print()
    {
        cout << "x : " << x << ", y : " << y << endl;
    }
};
```

생성자 오버로딩

```
int main(void)
{
    Point p1;
    Point p2(3, 5);

    p1.Print();
    p2.Print();

    return 0;
}
```

```
[_@localhost study]$ a.out
x : 0, y : 0
x : 3, y : 5
```

소멸자

- 객체가 소멸될 때 소멸자가 반드시 호출
- 소멸자 이름은 ~클래스 이름과 같음
ex) ~Point()
- 반환형이 없고, 어느 것도 반환하지 않음
- 매개 변수 없음 -> 오버로딩 불가능

소멸자

```
class IntArray
{
    int* arr;
    int len;
public:
    IntArray(int l)
    {
        cout << "생 성 " << endl;
        len = l;
        arr = new int[len];
    }
    int& GetEl(int i)
    {
        return arr[i];
    }
};
```

```
void Print()
{
    for(int i = 0; i < len; i++)
        cout << arr[i] << ' ';
    cout << endl;
}
~IntArray()
{
    cout << "소 멸 " << endl;
    delete[] arr;
}
};
```

소멸자

```
int main(void)
{
    IntArray arr(4);

    for(int i = 0; i < 4; i++)
    {
        cout << i + 1 << "번째 원소 입력 : ";
        cin >> arr.GetEl(i);
    }

    arr.Print();

    return 0;
}
```

[localhost study]\$ a.out

생성

1번째 원소 입력 : 4

2번째 원소 입력 : 9

3번째 원소 입력 : 1

4번째 원소 입력 : 2

4 9 1 2

소멸

디폴트 생성자, 소멸자

- 생성자를 정의하지 않은 경우, 디폴트 생성자가 동작
`Point() {}`
- 소멸자를 정의하지 않은 경우, 디폴트 소멸자가 동작
`~Point() {}`
- 생성자를 하나라도 정의한 경우, 디폴트 생성자는 사라짐
- 소멸자를 정의하면 디폴트 소멸자는 사라짐

디폴트 생성자, 소멸자

```
class Point
{
    int x, y;
public:
    Point(int a, int b)
    {
        x = a;
        y = b;
    }
};

int main(void)
{
    Point p1(3, 4);
    Point p2;
    return 0;
}
```

오류 : 해당하는 생성자 없음

멤버 초기화 구문

```
class Point
{
    int x, y;
public:
    Point(int a = 0, int b = 0) : x(a), y(b) {}

    void Print()
    {
        cout << "x : " << x << ", y : " << y << endl;
    }
};

int main(void)
{
    Point p1;
    Point p2(3, 4);

    p1.Print();
    p2.Print();

    return 0;
}
```

메모리 공간 생성과 동시에 초기화

```
int a;
a = 3;
```

```
int a = 3;
```

멤버 함수 외부 정의

```
class Point
{
    int x, y;
public:
    Point(int a = 0, int b = 0);
    void Print();
};

Point::Point(int a, int b) : x(a), y(b) {}

void Point::Print()
{
    cout << "x : " << x << ", y : " << y << endl;
}

int main(void)
{
    Point p1;
    Point p2(3, 4);

    p1.Print();
    p2.Print();

    return 0;
}
```