

민선생코딩학원 훈련반

수업노트 LV-18



배우는 내용

1. isPattern(index)로 패턴찾기
2. DAT(Direct Address Table)

배열에 Pattern 존재 여부 판단 문제

vect	B	T	A	B	C	Q	A	B	C
pattern	A	B	C						

- ▶ Pattern이 vect안에 존재할까?
→ vect의 2번, 6번 index에 존재
- ▶ Pattern이 존재하는지 확인하는 두가지 방법을 소개
 1. Flag를 쓰는 방법
 2. 함수를 쓰는 방법

isPattern 함수 사용하기

vect

B	T	A	B	C	Q	A	B	C
---	---	---	---	---	---	---	---	---

pattern

A	B	C
---	---	---

다른글자가
하나라도 발견되면
바로 return 0

```
#include <iostream>
using namespace std;

char vect[10] = "BTABCQABC";
char pattern[4] = "ABC";

int isPattern(int index)
{
    for (int x = 0; x < 3; x++)
    {
        if (pattern[x] != vect[x + index]) return 0;
    }
    return 1;
}

int main()
{
    int result;

    for (int x = 0; x <= 6; x++)
    {
        result = isPattern(x);
        if (result == 1) break;
    }

    if (result == 1) cout << "발견";
    else cout << "미발견";

    return 0;
}
```

[주의] for문을 0 ~ 6 까지 돌려야함
7까지 돌리면 Runtime Error 발생

Flag를 써서 푸는 Pattern 문제

vect **B** **T** **A** **B** **C** **Q** **A** **B** **C**

pattern **A** **B** **C**

- ▶ 함수를 쓰는 것이 구현하기 더 쉽다
- ▶ 복잡한 for문 일때 오른쪽 처럼 짜는 것 보다 함수를 사용 하는 것이 더 쉽게 코딩할 수 있다

flag를 이용해서
완전히 동일한지 판단

```
#include <iostream>
using namespace std;

char vect[10] = "BTABCQABC";
char pattern[4] = "ABC";

int main()
{
    int result;
    int flag;
    int y;

    for (int x = 0; x <= 6; x++)
    {
        flag = 0;
        for (y = 0; y < 3; y++)
        {
            if (pattern[y] != vect[x + y])
            {
                flag = 1;
                break;
            }
        }

        if (flag == 0) break;

        if (flag == 0) cout << "발견";
        else cout << "미발견";

        return 0;
    }
}
```

2차배열에서 패턴 찾기

- ▶ pattern이 vect에 존재하는지 찾는 문제
- ▶ 출력결과 : 발견

vect

0	3	5	1
1	2	4	5
2	3	6	1
3	3	1	1
4	4	5	6
5	2	4	5

pattern

2	4	5
---	---	---

```
#include <iostream>
using namespace std;

int vect[6][3] =
{
    3, 5, 1,
    2, 4, 5,
    3, 6, 1,
    3, 1, 1,
    4, 5, 6,
    2, 4, 5
};

int pattern[3] = { 2, 4, 5 };

int isPattern(int line)
{
    for (int x = 0; x < 3; x++)
    {
        if (pattern[x] != vect[line][x]) return 0;
    }
    return 1;
}
```

```
int main()
{
    int result;

    for (int y = 0; y <= 5; y++)
    {
        result = isPattern(y);
        if (result == 1) break;
    }

    if (result == 1) cout << "발견";
    else cout << "미발견";

    return 0;
}
```

2차배열에서 패턴 찾기

- ▶ pattern이 vect의 어느 좌표에 있는지 찾는 문제
- ▶ 출력결과 : 0,2 2,1

vect	1	2	3	4	1
	3	1	0	0	1
	2	3	4	1	2

pattern	3	4	1
---------	---	---	---

```
int isPattern(int dy, int dx)
{
    for (int x = 0; x < 3; x++)
    {
        if (pattern[x] != vect[dy][dx + x]) return 0;
    }
    return 1;
}
```

```
int main()
{
    int result;
    int x, y;

    for (y = 0; y < 3; y++)
    {
        for (x = 0; x <= 2; x++)
        {
            result = isPattern(y, x);
            if (result == 1) cout << y << ", " << x << endl;
        }
    }

    return 0;
}
```

[주의] for문 돌리는
영역이 0~2 까지만
돌려야함

Direct Addressing Table 이란?

- ▶ 값을 index로 활용하는 코딩 방식을 DAT(Direct Addressing Table)라고 함
- ▶ 정확한 이해를 위한 문제 예제

```
int bucket[200] = { 0 };  
char target = 'A';  
bucket[target] = 1;
```

배열에 어떤 종류의 알파벳들이 있는지 찾아내는 프로그램을 작성하시오

A	D	B	F	A	D
---	---	---	---	---	---

정답 : ABDF

Direct Addressing Table 예제1

A D B F A D

정답 : ABDF

문자를 index로 사용,
bucket[문자] = 1 을 넣는다

int bucket[200];

	⋮
65 (A)	✓
66 (B)	✓
67 (C)	
68 (D)	✓
69 (E)	
70 (F)	✓
	⋮

```
#include <iostream>
using namespace std;

int main()
{
    int bucket[200] = { 0 };
    char vect[7] = "ADBFAD";

    for (int x = 0; x < 6; x++)
    {
        bucket[vect[x]] = 1;
    }

    for (int x = 0; x < 200; x++)
    {
        if (bucket[x] == 1) cout << (char)(x);
    }

    return 0;
}
```

Direct Addressing Table 예제2

- ▶ 각각 숫자가 몇 개인지 출력하기

4	1	1	1	5	4
---	---	---	---	---	---

출력결과

1:3개

4:2개

5:1개

```
#include <iostream>
using namespace std;

int main()
{
    int bucket[200] = { 0 };
    int vect[6] = { 4, 1, 1, 1, 5, 4 };

    for (int x = 0; x < 6; x++)
    {
        bucket[vect[x]]++;
    }

    for (int x = 0; x < 200; x++)
    {
        if (bucket[x] > 0)
        {
            cout << x << " : " << bucket[x] << "개\n";
        }
    }

    return 0;
}
```