

프로그래밍 역량 강화 전문기관, 민코딩

---

# 함수 만들기



# 배우는 내용

## 함수 만들기

1. for문 추가 반복자 활용
2. 문자형 변수 char
3. 함수로 명령어 만들기

# 배열에 값 채우기 1

다섯 칸의 배열을 선언하고, for 문으로 3 부터 7 까지 채우기

3	4	5	6	7
---	---	---	---	---

```
int arr[5];  
int x;  
  
for (x=0; x<5; x++)  
{  
    arr[x] = ?  
}
```

이곳에 어떤 값을 넣어야  
배열에 3부터 7까지 대입될까?

## 배열에 값 채우기 2

```
int arr[5];  
int x;  
  
int t = 3;  
for (x=0; x<5; x++)  
{  
    arr[x] = t;  
    t++;  
}
```

arr

3	4	5	6	7
---	---	---	---	---

변수를 하나 더 사용하면, 문제를 쉽게 해결할 수 있다.

# 반복자 2개 활용하기 1

다섯 칸의 배열에 25 ~ 21 값 채우기

25	24	23	22	21
----	----	----	----	----

```
int arr[5];  
int x;
```

```
for (x=0; x<5; x++)  
{  
    arr[x] = ?  
}
```

for를 0부터 4까지 돌렸다.  
어떻게 해야 25부터 21까지 값이 들어갈까?

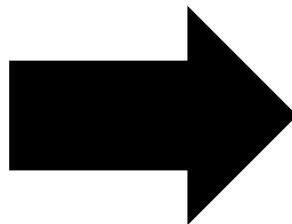
# 반복자 2개 활용하기 2

다섯 칸의 배열에 25 ~ 21 값 채우기

25	24	23	22	21
----	----	----	----	----

```
int arr[5];
int x;

for (x=0; x<5; x++)
{
    arr[x] = ?
}
```



```
int arr[5];
int x;

int t = 25;
for (x=0; x<5; x++)
{
    arr[x] = t;
    t--;
}
```

역시 변수 t를 활용하면  
쉽게 문제를 풀 수 있다.

# 반복자 2개 활용하기 3

방금 문제를 풀기 위해 **for문을 25부터 21까지** 돌린다면?

25	24	23	22	21
----	----	----	----	----

```
int arr[5];  
int x;  
  
for (x=25; x>=21; x--)  
{  
    arr[?] = x;  
}
```

이번에는 for 문을 25부터 21까지 돌렸다.  
어떻게 값을 채워야 할까?

# 반복자 2개 활용하기 4

방금 문제를 풀기 위해 **for문을 25부터 21까지** 돌린다면?

25	24	23	22	21
----	----	----	----	----

```
int arr[5];  
int x;  
  
int t = 0;  
for (x=25; x>=21; x--)  
{  
    arr[ t ] = x;  
    t++;  
}
```

index 값을 0부터 4까지 만들어야 한다.  
변수 t를 쓰면 간단히 해결 가능하다.

추가 반복자를 써서 for문을 돌리는 방식은  
자주 쓰는 기법이므로, 꼭 연습해두자.

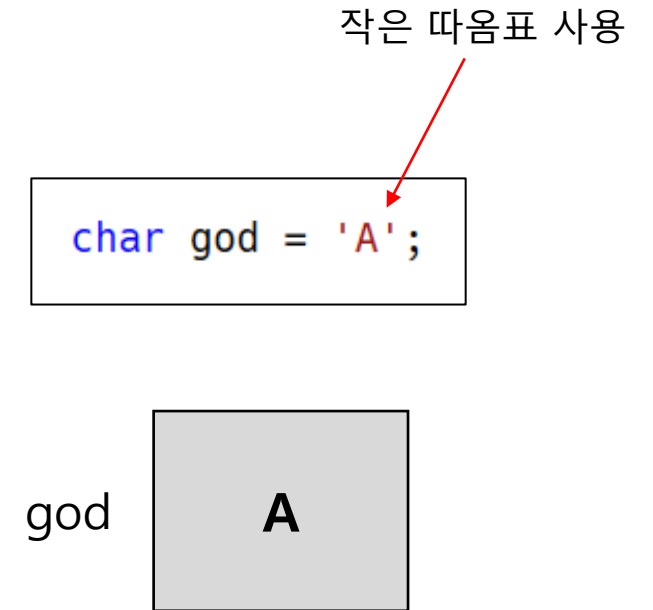
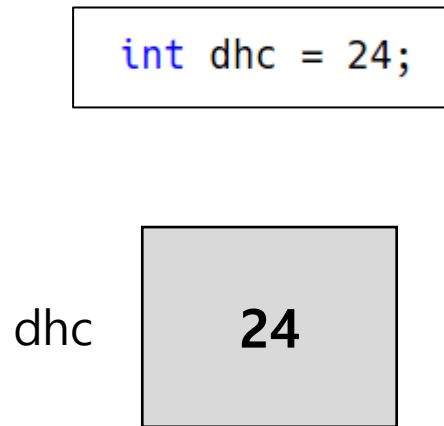


# char 변수란?

char 타입의 변수는 **하나의 문자**를 저장할 수 있다. (int 타입의 변수는 숫자를 저장 함)

**캐릭터 변수**라고 읽는다.

- char를 "찰"이라고 부르지 말자.



# char 배열 하드코딩 방법

char arr[4] 배열에 문자 a, b, c, d를 하드코딩하는 두 가지 방법

```
char arr[4] = {'a', 'b', 'c', 'd'};
```

```
char arr[4];  
arr[0]='a';  
arr[1]='b';  
arr[2]='c';  
arr[3]='d';
```

# 문자 2개 입력 받고 출력하기

```
#include <iostream>
using namespace std;
int main()
{
    char a, b;

    cin >> a >> b;

    cout << a << b;

    return 0;
}
```

int형 변수와 사용법과 유사하다.

char형을 쓸 때는 문자와 숫자를 잘 구분하자.

→ int형과 char형을 헷갈리지 않도록 주의해야 한다.

# char 배열에 for문 사용하기

## [문제]

ch 변수에 문자를 입력 받는다.

for 문을 돌려 배열에 ch 값으로 채우자.

for 문을 돌려 배열의 값들을 출력하자.

```
int arr[4];
int x;
char ch;

cin >> ch;

for (x=0; x<4; x++)
{
    arr[x] = ch;
}

for (x=0; x<4; x++)
{
    cout << arr[x];
}
```

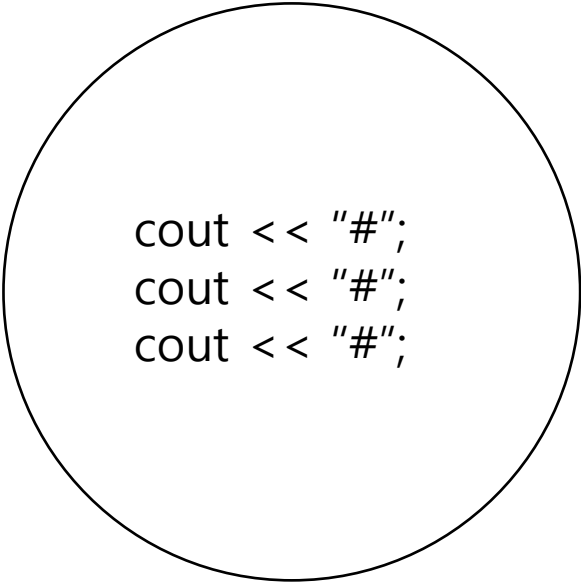
입력 : 5

출력 : 5555

# 함수란?

함수는 소스코드를 묶어서 **나만의 명령어를 만드는 것**이다.

BBQ 명령어



```
cout << "#";  
cout << "#";  
cout << "#";
```

```
#include <iostream>  
using namespace std;  
int main()  
{  
    BBQ();  
    BBQ();  
  
    return 0;  
}
```

실행결과 : #####

# 함수의 실제 소스코드

함수(명령어)를 사용하기 위해서는, 먼저 함수를 만들어야 한다.

main에서는 BBQ함수를 두 번 수행한다.

BBQ함수를 두 번 “호출한다” 라고 표현한다.

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

void BBQ( )
{
    cout << "#";
    cout << "#";
    cout << "#";
}

int main( )
{
    BBQ( );
    BBQ( );

    return 0;
}
```

# main도 함수이다.

C, C++, Java, C# 등의 언어들 모두

소스코드에 반드시 main 함수가 존재해야 한다.

main함수는 프로그램을 실행 하자마자 처음 수행되는 함수이다.

→ 모든 프로그램은 **main함수에서 시작하고,**  
**main함수에서 프로그램이 종료된다.**

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

void BBQ( )
{
    cout << "#";
    cout << "#";
    cout << "#";
}

int main( )
{
    BBQ( );
    BBQ( );

    return 0;
}
```

# [중요] 함수 소스코드 이해하기

두 소스코드를 이해 해보자.

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

void timer()
{
    cout << "START TIME" << endl;

    int x;
    for (x = 0; x < 5; x++)
    {
        cout << x << " ";
    }

    cout << endl;
    cout << "END TIME" << endl;
}

int main()
{
    timer();

    return 0;
}
```

## 실행결과

START TIME  
0 1 2 3 4  
END TIME

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

void fire()
{
    cout << "FIRE";
}

void water()
{
    cout << ">>>";
    fire();
    cout << "<<<";
    cout << endl;
}

int main()
{
    cout << "1" << endl;
    water();

    cout << "2" << endl;
    water();

    return 0;
}
```

## 실행결과

1  
>>>FIRE<<<  
2  
>>>FIRE<<<



# 함수의 위치

main함수에서 RIDE 함수를 호출하려면  
**main함수 위에** 함수가 존재해야 한다.

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

void RIDE()
{
    int t;

    cin >> t;
    cout << t;
}

int main()
{
    RIDE();

    return 0;
}
```

만약 main함수 아래에 있다면  
호출하지 못하고 **Compile Error**가 발생한다.

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

int main()
{
    RIDE();

    return 0;
}

void RIDE()
{
    int t;

    cin >> t;
    cout << t;
}
```

# 컴파일 에러 발생 이유 찾아보기

컴파일 에러가 나는 소스코드

```
void kfc()
{
    bbq();
}

void bbq()
{

}

int main()
{
    kfc();
    return 0;
}
```

bbq함수를 호출하기 위해서  
kfc함수보다 위에 bbq함수를 만들어야 한다.

# 지역 변수란?

✓ 함수 안에 만들어지는 일반 변수들을 **지역 변수**라고 한다.

- 우리가 main함수에서 만들던 변수들도 모두 지역 변수이다.

✓ abc 함수 안에 만든 변수 t와  
main 함수 안에 만든 변수 t는  
**이름만 같을 뿐, 완전히 다른 변수이다!**

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

```
void abc()
{
    int t = 50;
}
```

```
int main()
{
    int t = 100;
    abc();
    cout << t ;

    return 0;
}
```

두개의 변수 t는 이름만 같을 뿐,  
완전히 다른 변수이기 때문에 100이 출력 된다.

# 전역 변수

함수 밖에 만든 변수를 **전역 변수**라고 한다.  
모든 함수들이 이 전역변수를 사용할 수 있다.

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

```
int t;
```

```
void abc()
{
    t = 50;
    cout << t;
}
```

```
int main()
{
    t = 100;
    abc();
    cout << t;

    return 0;
}
```

main 함수에서 int t에 100이 대입된다.

이후, abc함수에서 t 변수가 50으로 바뀌어 출력된다.  
따라서 출력 결과는 5050 이다.

# 전역 변수 예제

✓ main 함수에서 input, output 함수를 호출 한다.

- input 함수에서 숫자 2개를 입력한다.
- output 함수에서는 input 함수에서 입력 받은 숫자 2개 출력된다.

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

int a, b;

void output()
{
    cout << a << b;
}

void input()
{
    cin >> a >> b;
}

int main()
{
    input();
    output();

    return 0;
}
```

# 전역 변수 사용시 주의사항

- ✓ 프로그래밍을 학습할 때는 전역변수를 마음대로 사용해도 된다.  
하지만 프로젝트의 규모가 커질 수록, 반드시 전역 변수가 필요할 때만 사용해야 한다.
  - 전역변수를 쓰지않고 프로그래밍하는 방법은 차후에 배운다.
- ✓ 전역 변수를 남용하면 **메모리 낭비가 심해진다**.
  - 메모리 낭비가 심한 프로그램을 실행시키려면 보다 높은 성능의 컴퓨터가 필요하다.

코딩 공부할 때는 상관없지만,  
실무에서는 꼭 필요한 경우에만 전역 변수를 사용한다.

# 전역변수 쓸 때 주의할 점

## ✓ for문 쓸 때, 지역변수로 for문을 돌리자.

- for문 돌릴 때 변수 x를 전역 변수로 쓴다면 버그가 발생한다.

## ✓ 버그가 발생하는 이유

1. main함수에서 x=0일때 abc함수를 호출 한다.
2. abc함수에서 변수 x를 이용하여 for문이 3번 반복 된다.
3. abc함수에서 for 이 끝날 때 x=3이 된 상태로 abc함수를 종료한다.
4. main으로 돌아왔을 때 x가 3이 되었기에 for문이 바로 종료된다.

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

```
int x;
```

```
void abc()
{
    for (x=0; x<3; x++)
    {
        cout << #;
    }
}
```

```
int main()
{
    for (x=0; x<3; x++)
    {
        abc();
    }
    return 0;
}
```

main에 있는 for문이 1 회 수행 후  
즉시 종료되는 버그가 발생