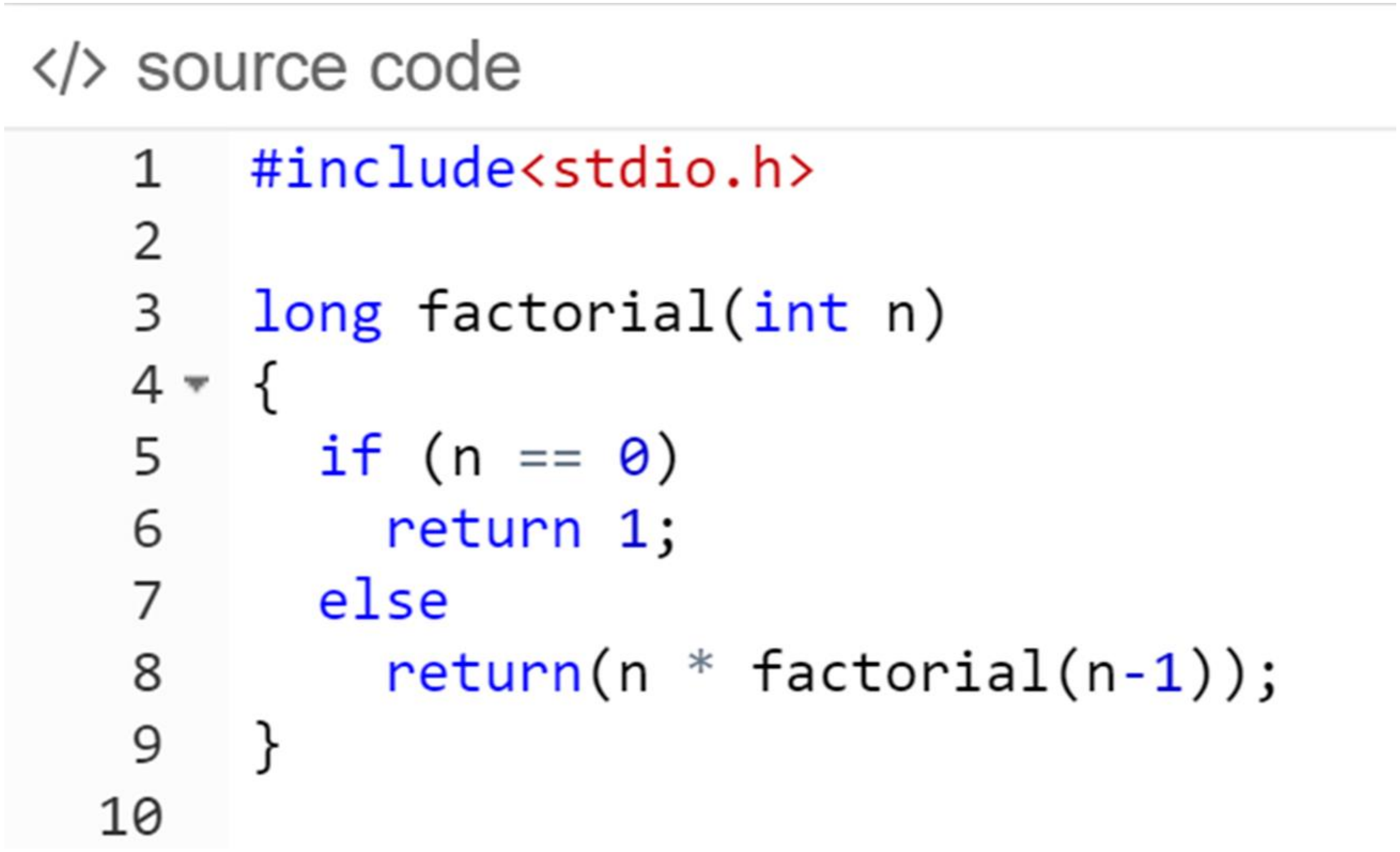
Programming with C and C++

CSC-101 (Lectures 19 and 20)

Dr. R. Balasubramanian
Professor
Department of Computer Science and Engineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee
Roorkee 247 667

bala@cs.iitr.ac.in
https://faculty.iitr.ac.in/cs/bala/

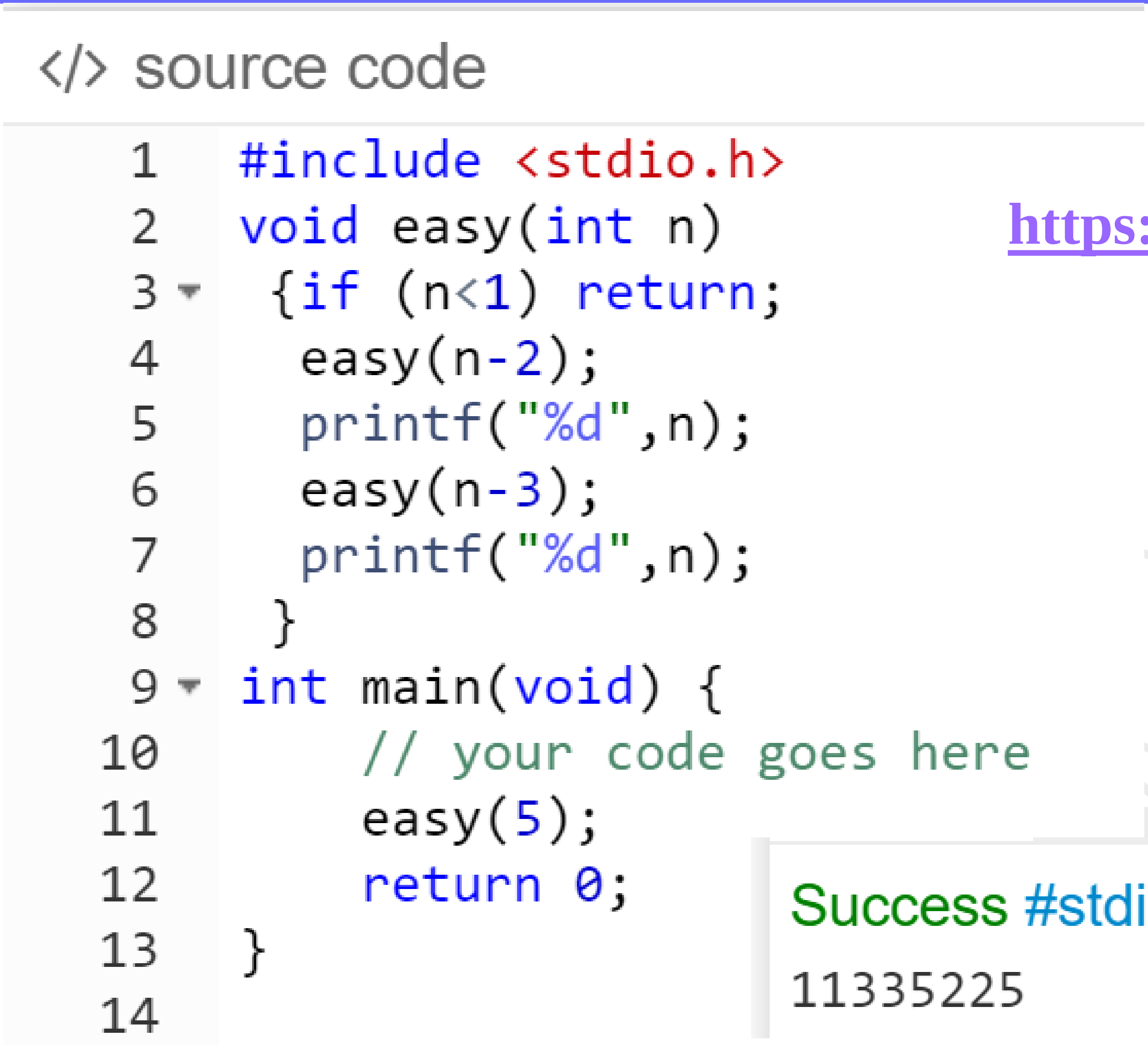
Stacks



</> source code

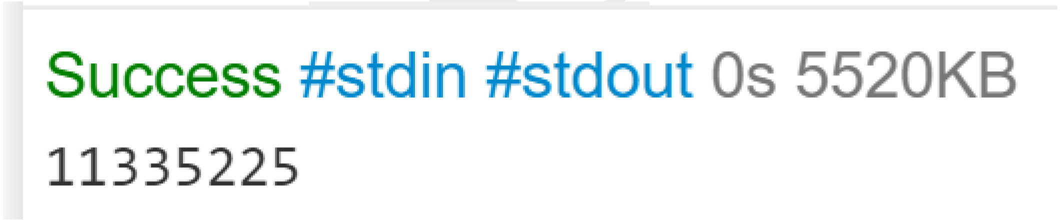
```c
#include<stdio.h>

long factorial(int n)
{
  if (n == 0)
    return 1;
  else
    return(n * factorial(n-1));
}

```

Stacks


</> source code

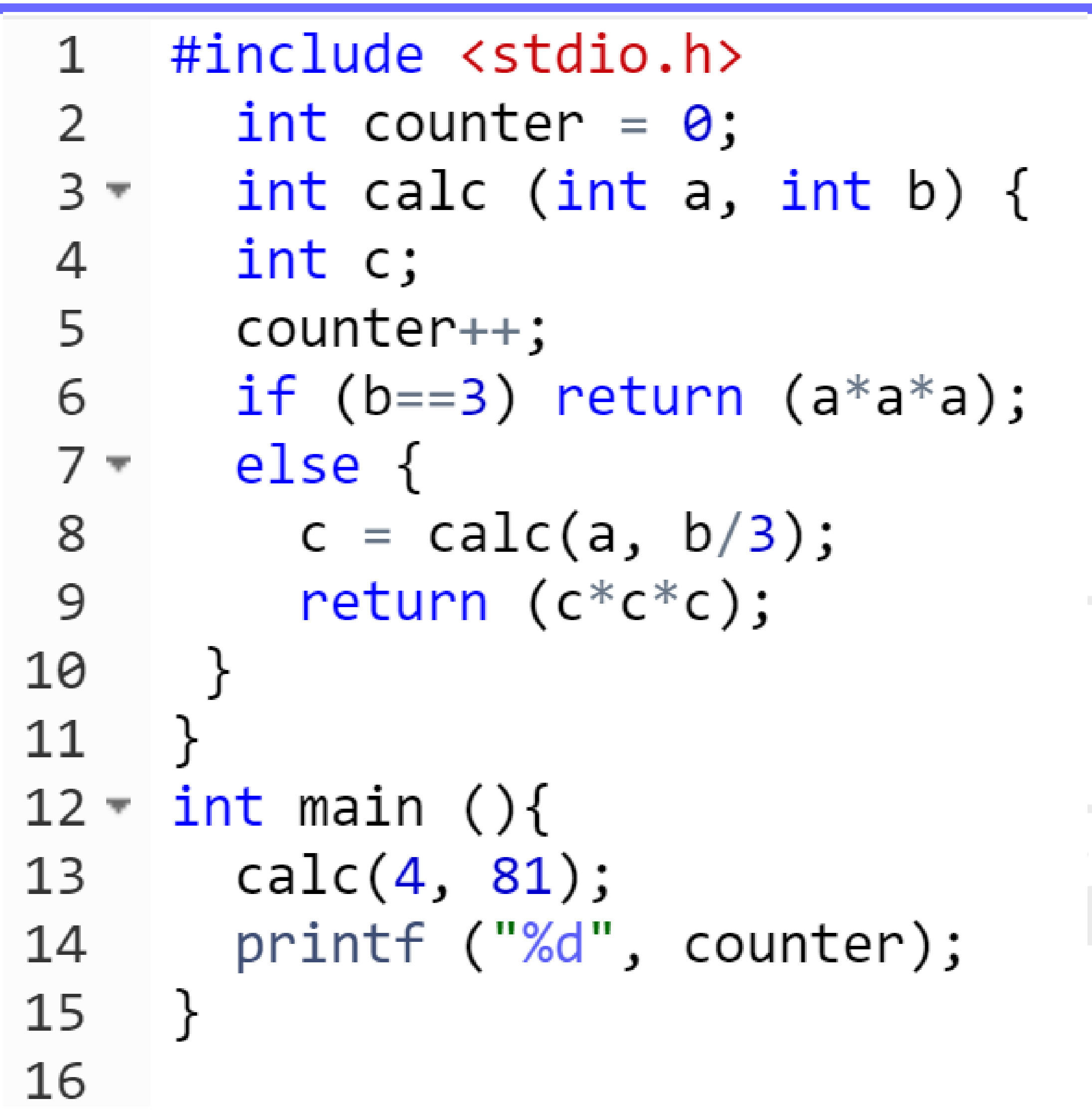
```
#include <stdio.h>
void easy(int n)
 {if (n<1) return;
   easy(n-2);
   printf("%d",n);
   easy(n-3);
   printf("%d",n);
 }
int main(void) {
	// your code goes here
	easy(5);
	return 0;
}
```

https://ideone.com/SPmToe


Success #stdin #stdout 0s 5520KB

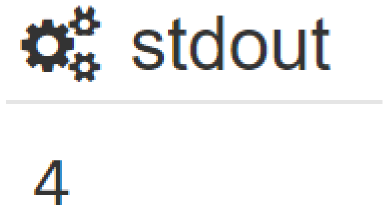
```
11335225
```

Recursion example



```c
#include <stdio.h>
  int counter = 0;
  int calc (int a, int b) {
  int c;
  counter++;
  if (b==3) return (a*a*a);
  else {
    c = calc(a, b/3);
    return (c*c*c);
 }
}
int main (){
  calc(4, 81);
  printf ("%d", counter);
}
```

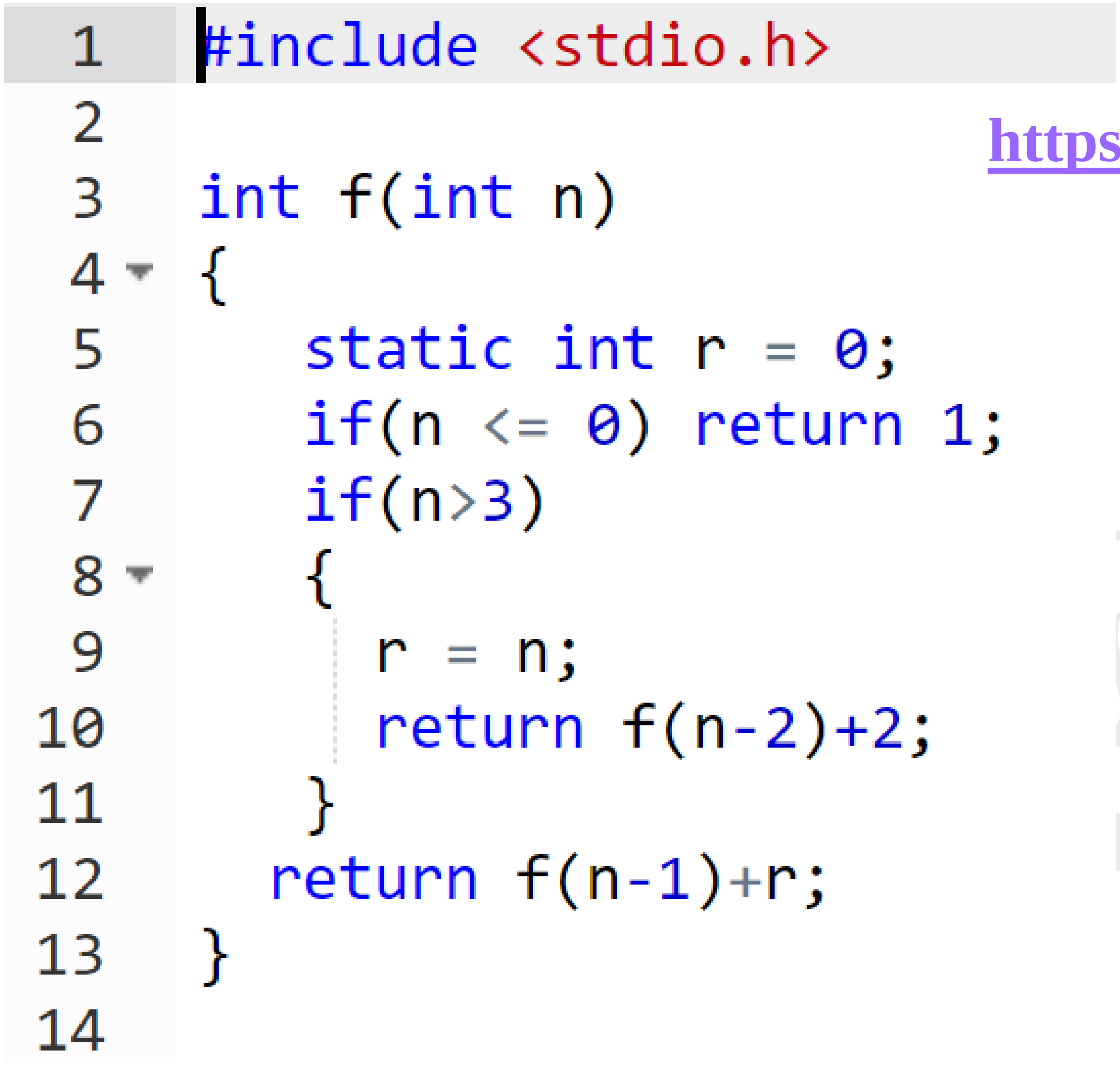
https://ideone.com/lI7cOC



stdout

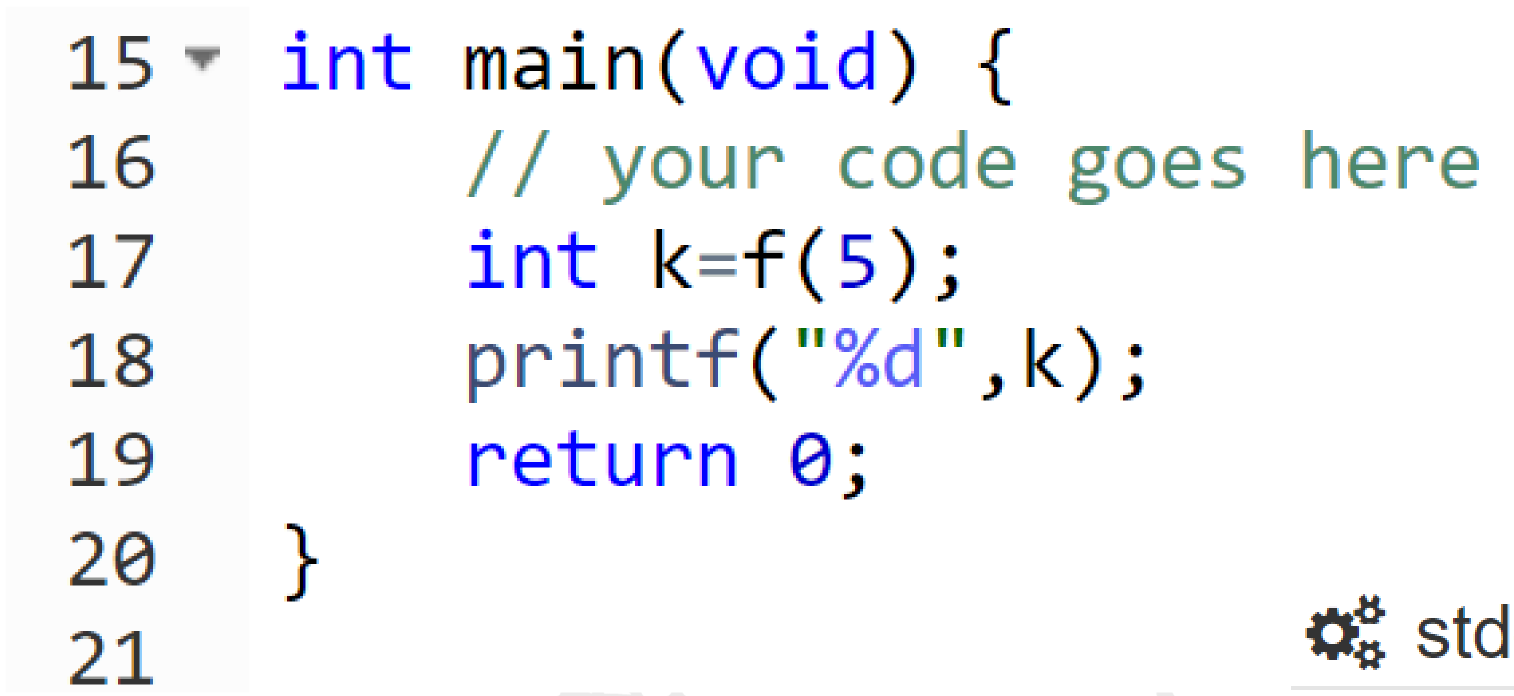
```
4
```

Recursion example

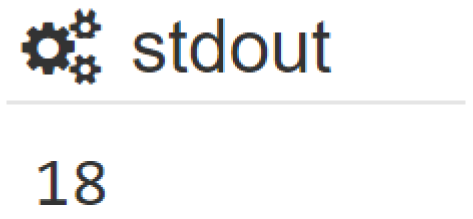


```c
#include <stdio.h>

int f(int n)
{
    static int r = 0;
    if(n <= 0) return 1;
    if(n>3)
    {
        r = n;
        return f(n-2)+2;
    }
   return f(n-1)+r;
}

```

https://ideone.com/t7HDTm



```c
int main(void) {
	// your code goes here
	int k=f(5);
	printf("%d",k);
	return 0;
}
```



```
stdout
18
```

Steps for Converting Iterative Code to Recursive

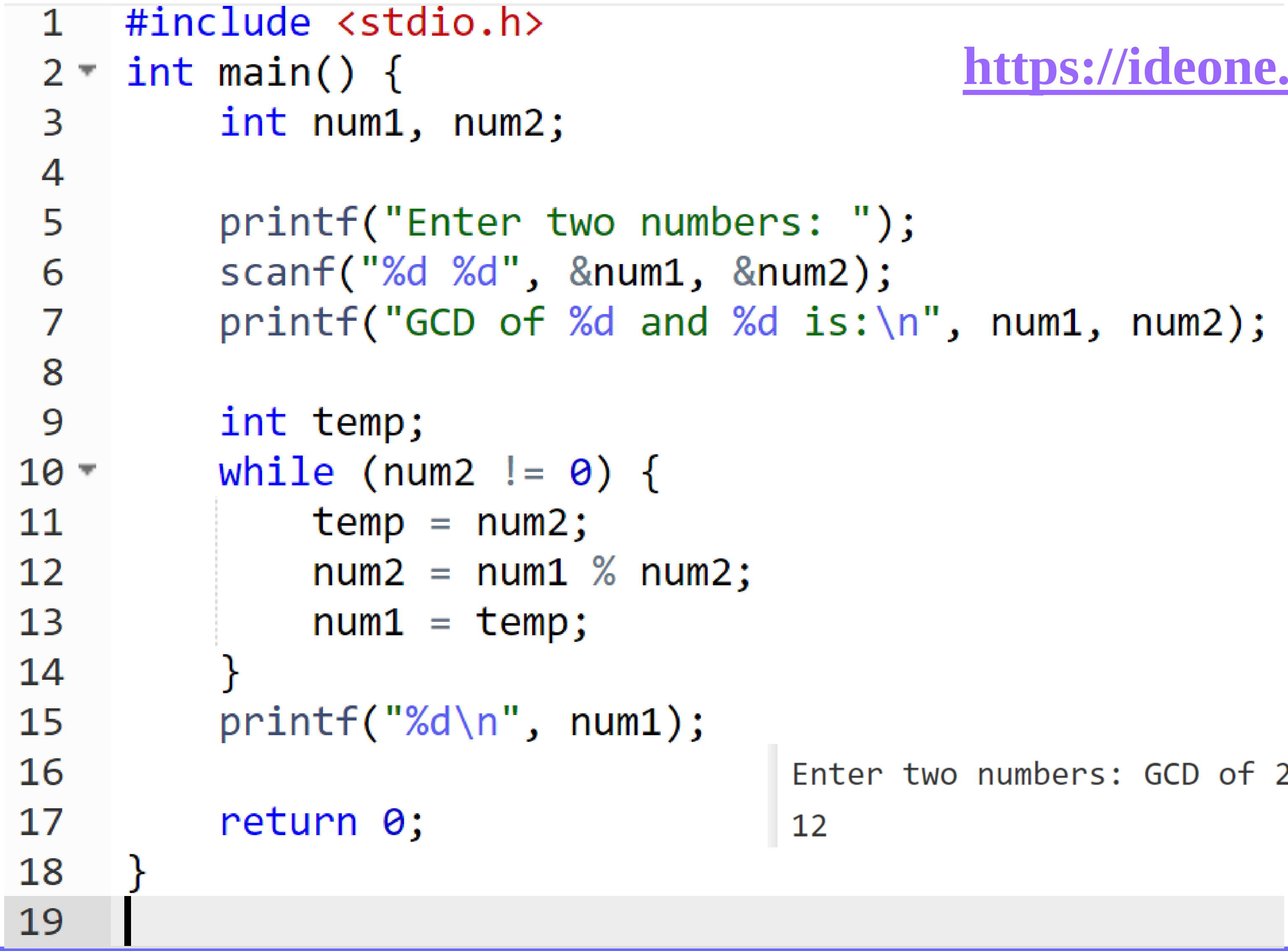
- Identify the main loop.
- Use the loop condition as the base case and the body of the loop as the recursive case.
- The local variables in the iterative version turn into the parameters of the recursive version.
- Compile and rerun tests.

Iteration to Recursion

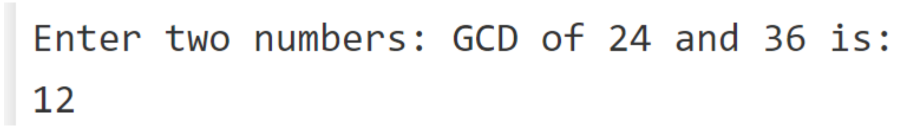
```
int sum=0;
for(int i=1;i<=100;++i){sum+=i;}

int GetTotal(int number)
{
   if (number==1) return 1;   //The end number
   return number+GetTotal(number-1); //The inner
// recursive
}

sum=1;
int GetTotal (int number, int sum)
{
    if(number==1) return sum;
    return GetTotal(number-1,sum+number);
}
```

Iteration

https://ideone.com/3NtrSa

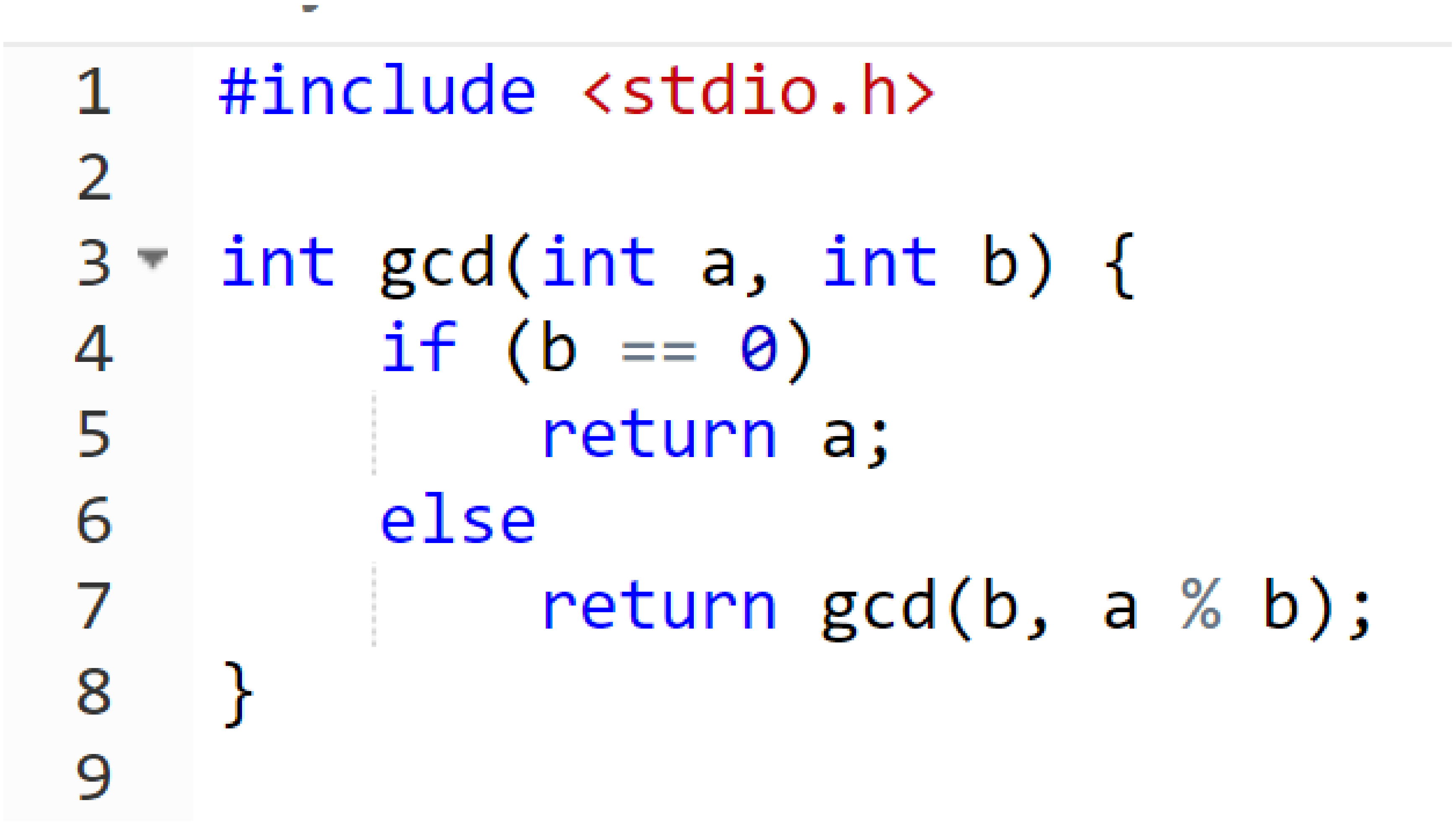


```c
#include <stdio.h>
int main() {
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("GCD of %d and %d is:\n", num1, num2);

    int temp;
    while (num2 != 0) {
        temp = num2;
        num2 = num1 % num2;
        num1 = temp;
    }
    printf("%d\n", num1);

    return 0;
}
```



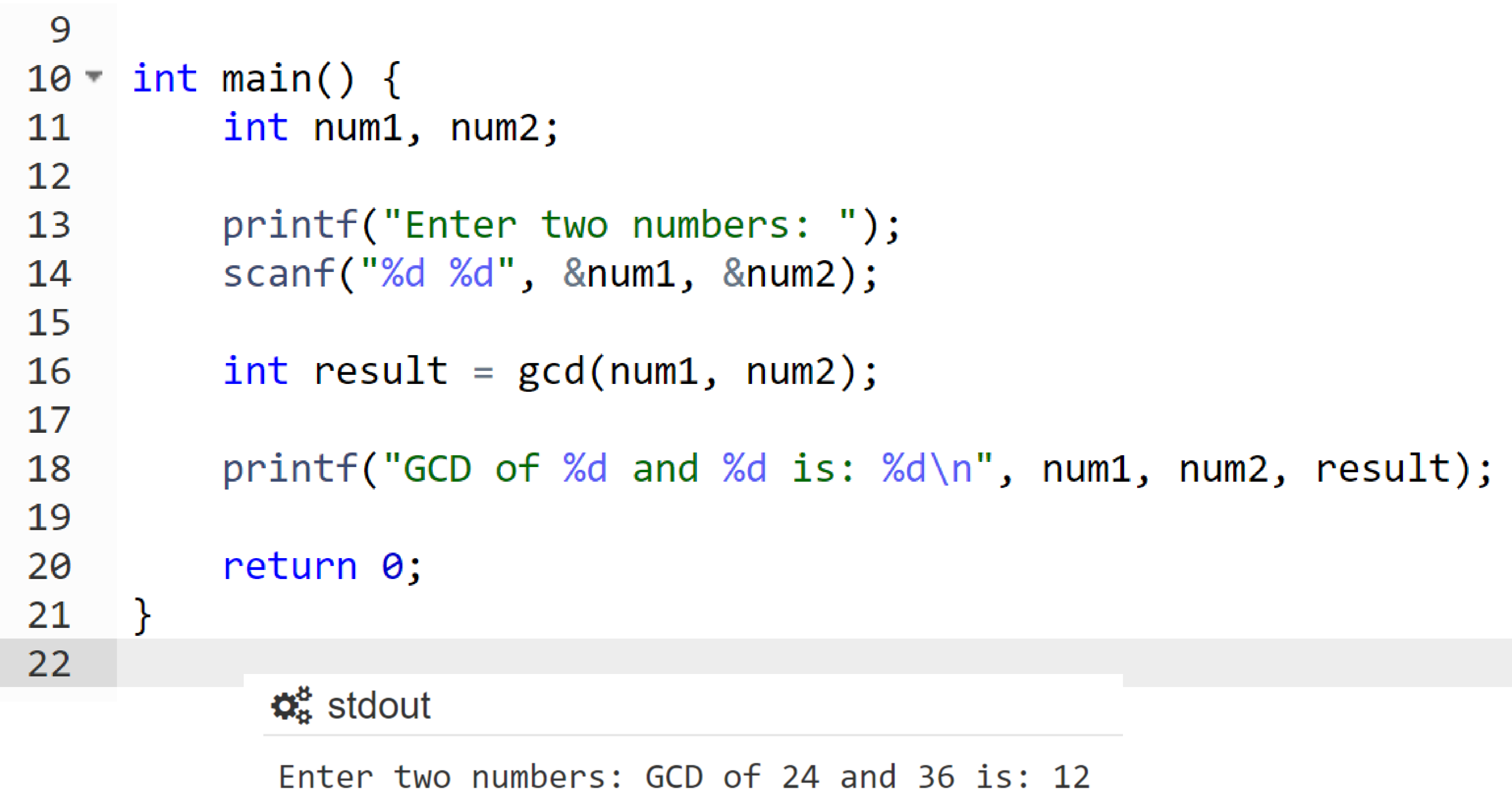
```
Enter two numbers: GCD of 24 and 36 is:
12
```

Recursion



```c
#include <stdio.h>

int gcd(int a, int b) {
	if (b == 0)
		return a;
	else
		return gcd(b, a % b);
}

```

https://ideone.com/sAnTHB



```c

int main() {
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    int result = gcd(num1, num2);

    printf("GCD of %d and %d is: %d\n", num1, num2, result);

    return 0;
}

```

stdout

```
Enter two numbers: GCD of 24 and 36 is: 12
```

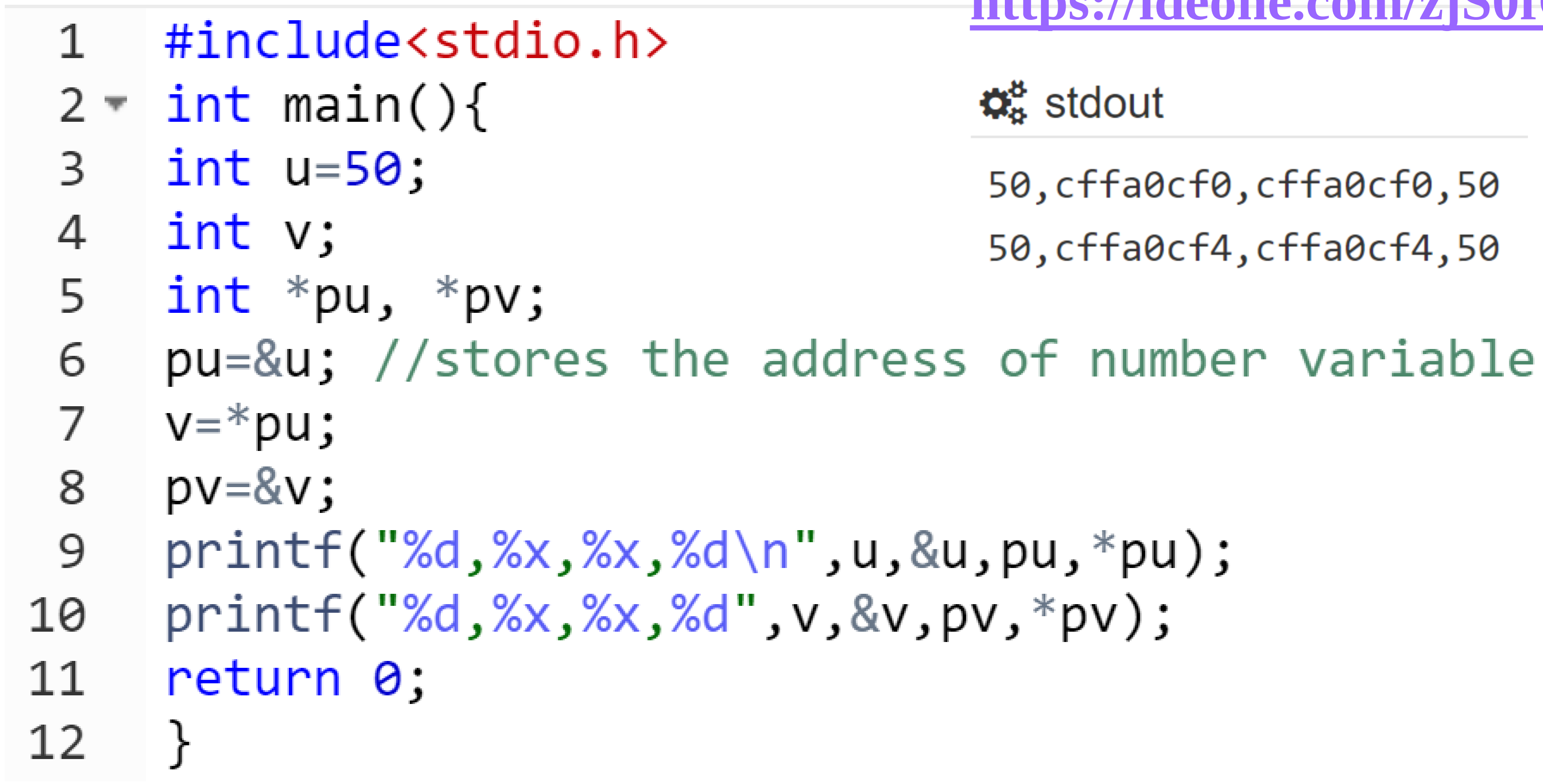
Recursion to Iteration

- Step 1: Identify the tail-recursive function in the algorithm.
- Step 2: Transform the function to use an iterative loop instead of recursive calls.
- Step 3: Replace the function arguments with variables that hold the intermediate state of the computation.
- Step 4: Update the variables in each iteration of the loop based on the problem's logic.
- Step 5: Repeat the loop until the base case is reached.

Dereference Operator (*) in C

- A pointer is a variable that points to an address of an another variable.

https://ideone.com/zjS0fC

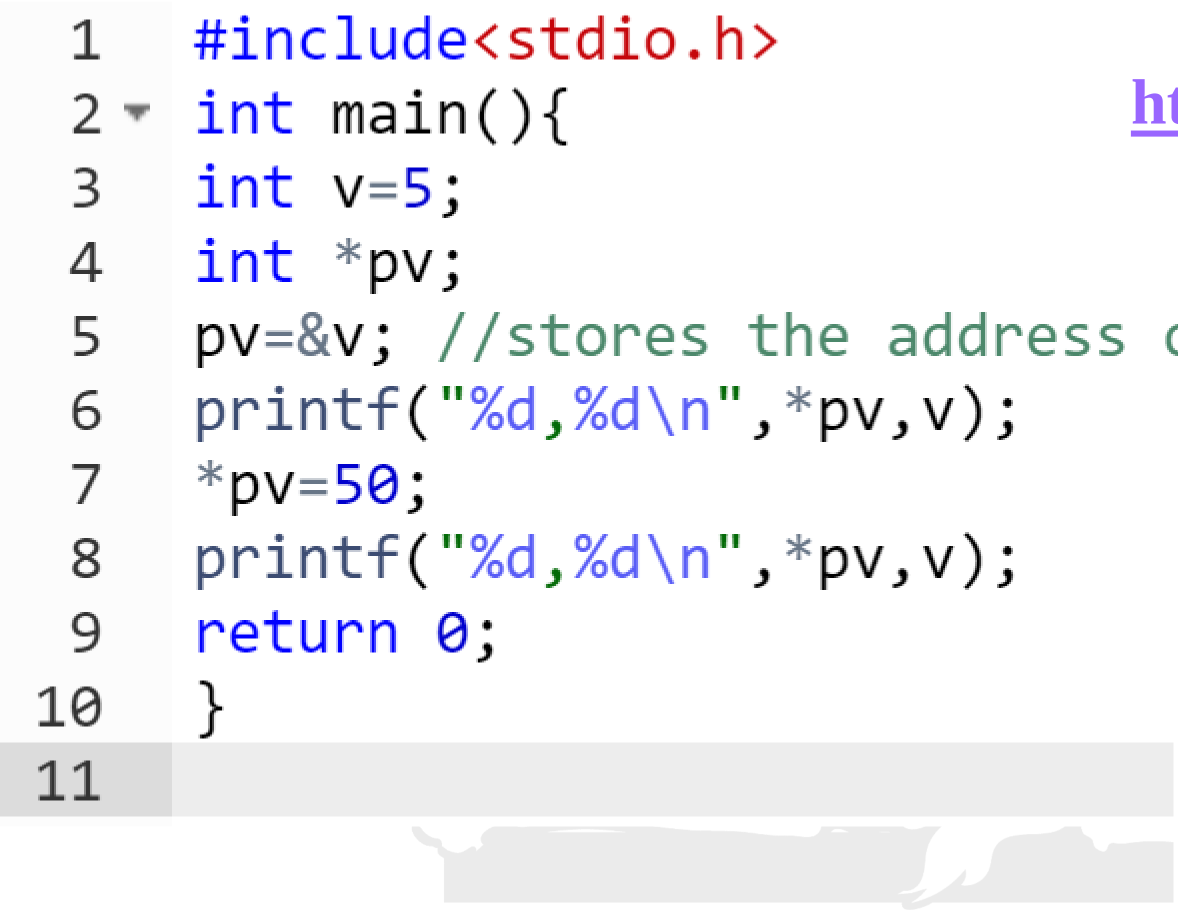


```c
#include<stdio.h>
int main(){
int u=50;
int v;
int *pu, *pv;
pu=&u; //stores the address of number variable
v=*pu;
pv=&v;
printf("%d,%x,%x,%d\n",u,&u,pu,*pu);
printf("%d,%x,%x,%d",v,&v,pv,*pv);
return 0;
}
```

stdout

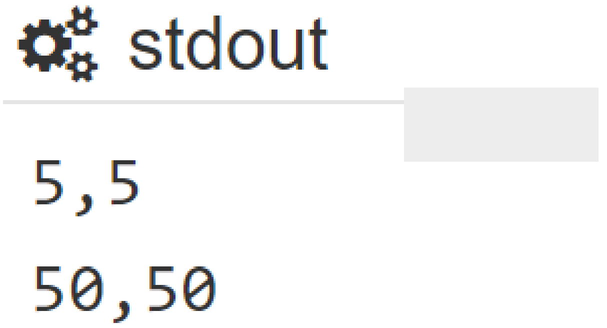
```
50,cffa0cf0,cffa0cf0,50
50,cffa0cf4,cffa0cf4,50
```

Pointers



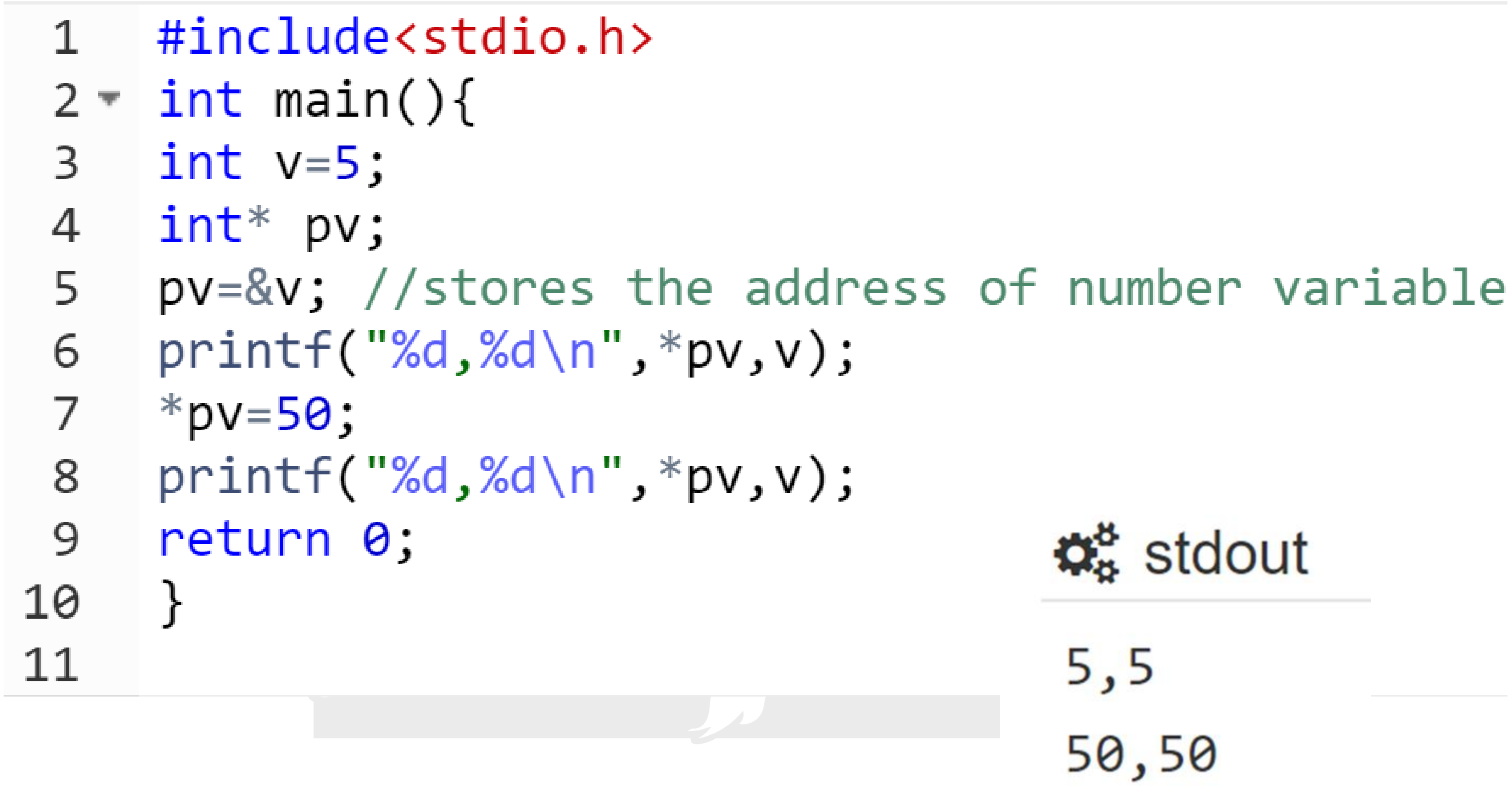
```c
#include<stdio.h>
int main(){
int v=5;
int *pv;
pv=&v; //stores the address of number variable
printf("%d,%d\n",*pv,v);
*pv=50;
printf("%d,%d\n",*pv,v);
return 0;
}
```

https://ideone.com/JNOVLJ



```
5,5
50,50
```

Pointers



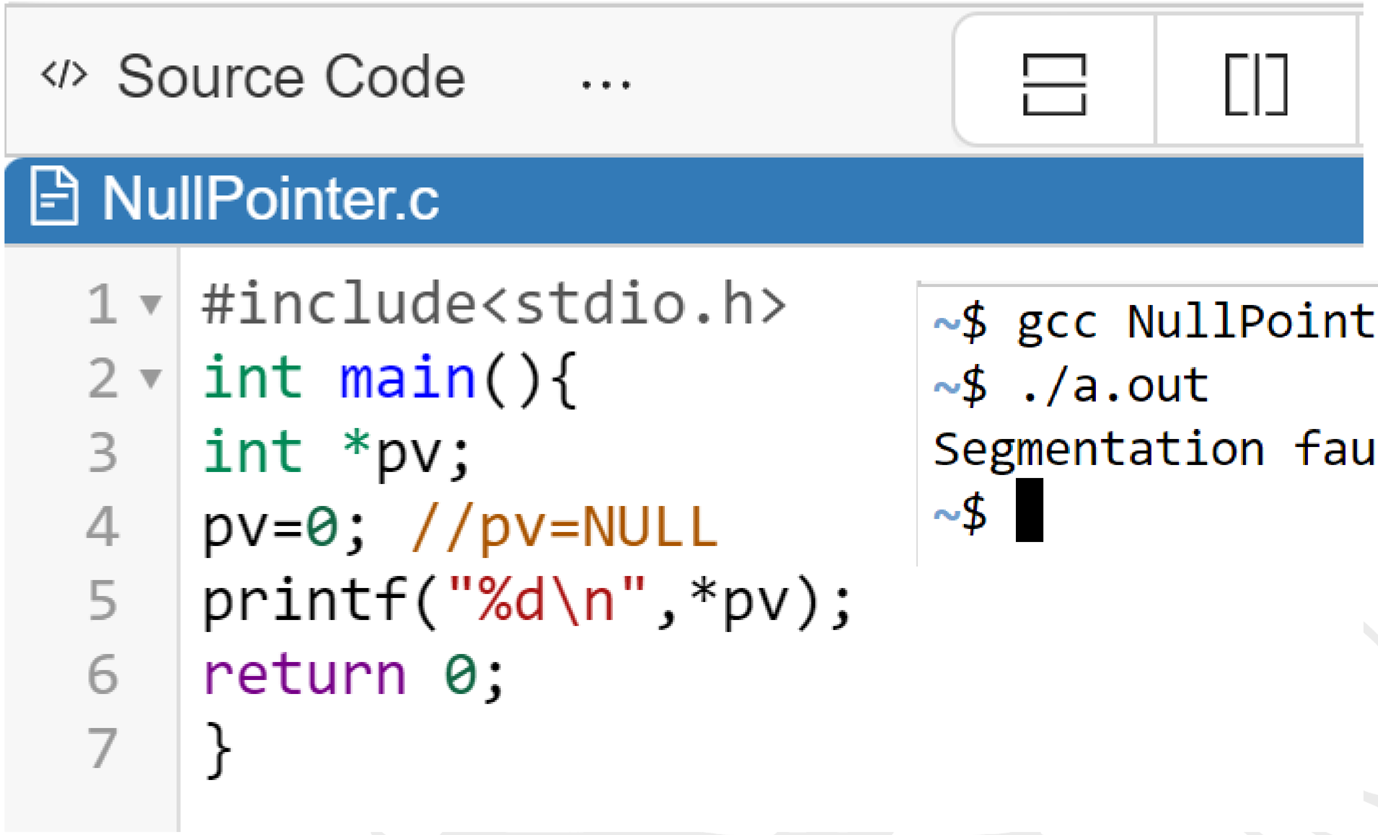
```
#include<stdio.h>
int main(){
int v=5;
int* pv;
pv=&v; //stores the address of number variable
printf("%d,%d\n",*pv,v);
*pv=50;
printf("%d,%d\n",*pv,v);
return 0;
}
```

stdout

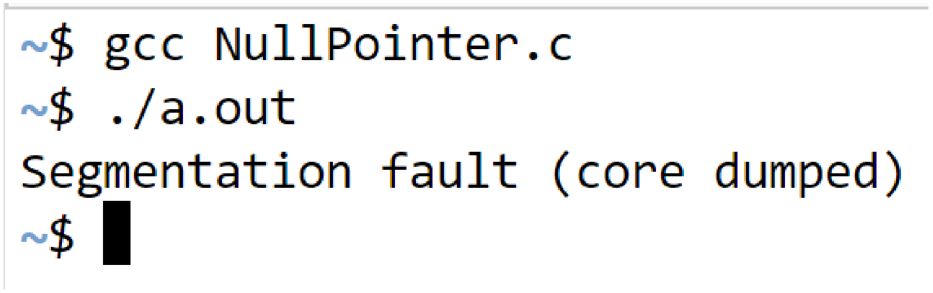
```
5,5
50,50
```

https://ideone.com/SzzM0D

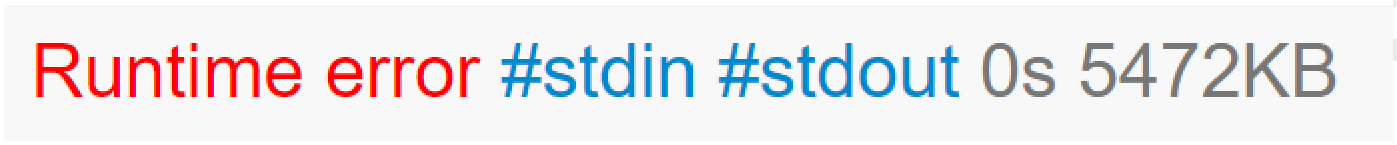
Null Pointer



```c
#include<stdio.h>
int main(){
int *pv;
pv=0; //pv=NULL
printf("%d\n",*pv);
return 0;
}
```



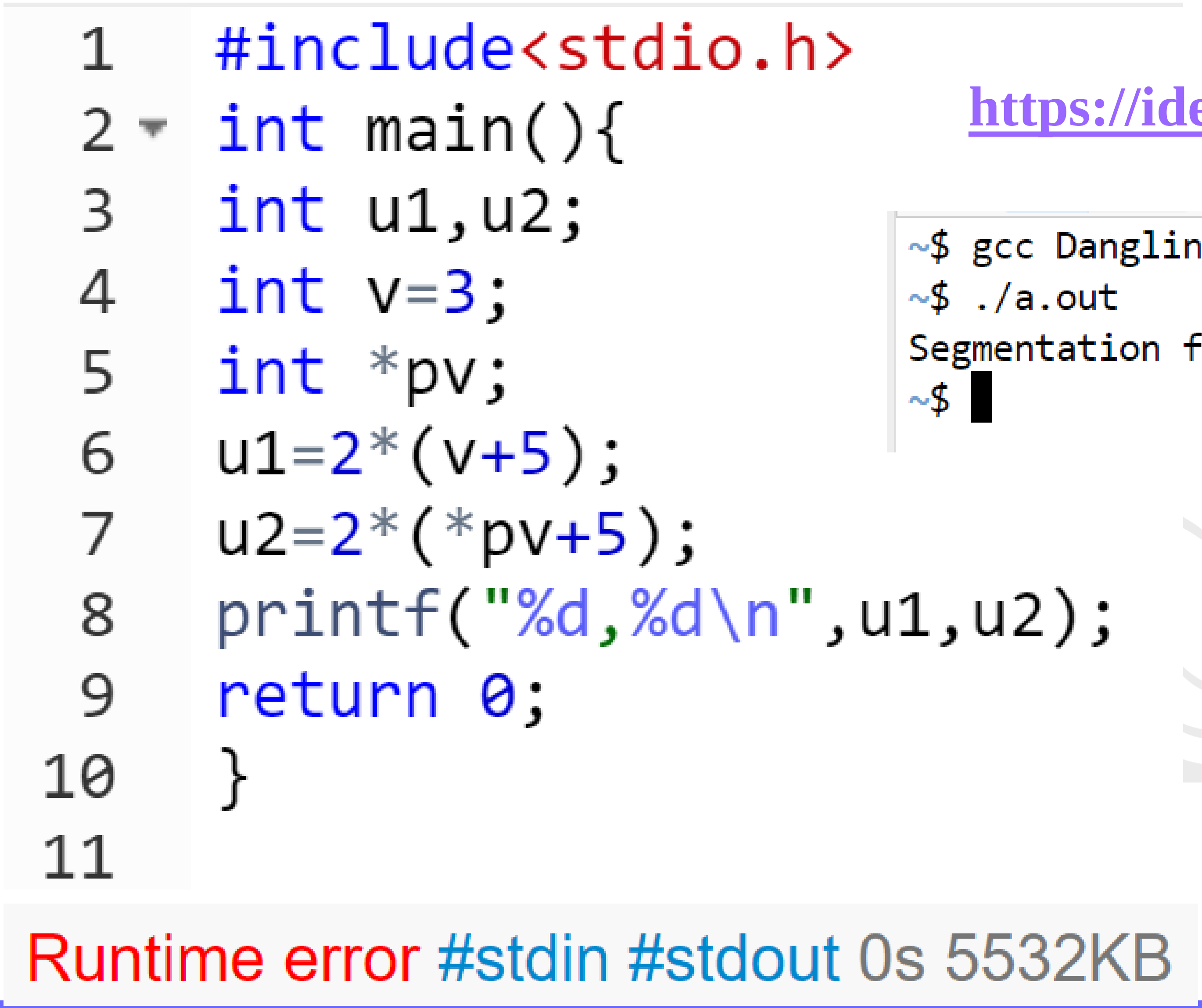
```
~$ gcc NullPointer.c
~$ ./a.out
Segmentation fault (core dumped)
~$
```



Runtime error #stdin #stdout 0s 5472KB

https://ideone.com/YCDKGg

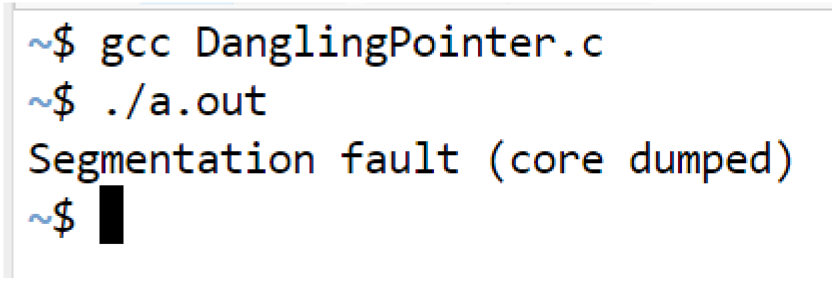
Dangling Pointer



```c
#include<stdio.h>
int main(){
int u1,u2;
int v=3;
int *pv;
u1=2*(v+5);
u2=2*(*pv+5);
printf("%d,%d\n",u1,u2);
return 0;
}

```

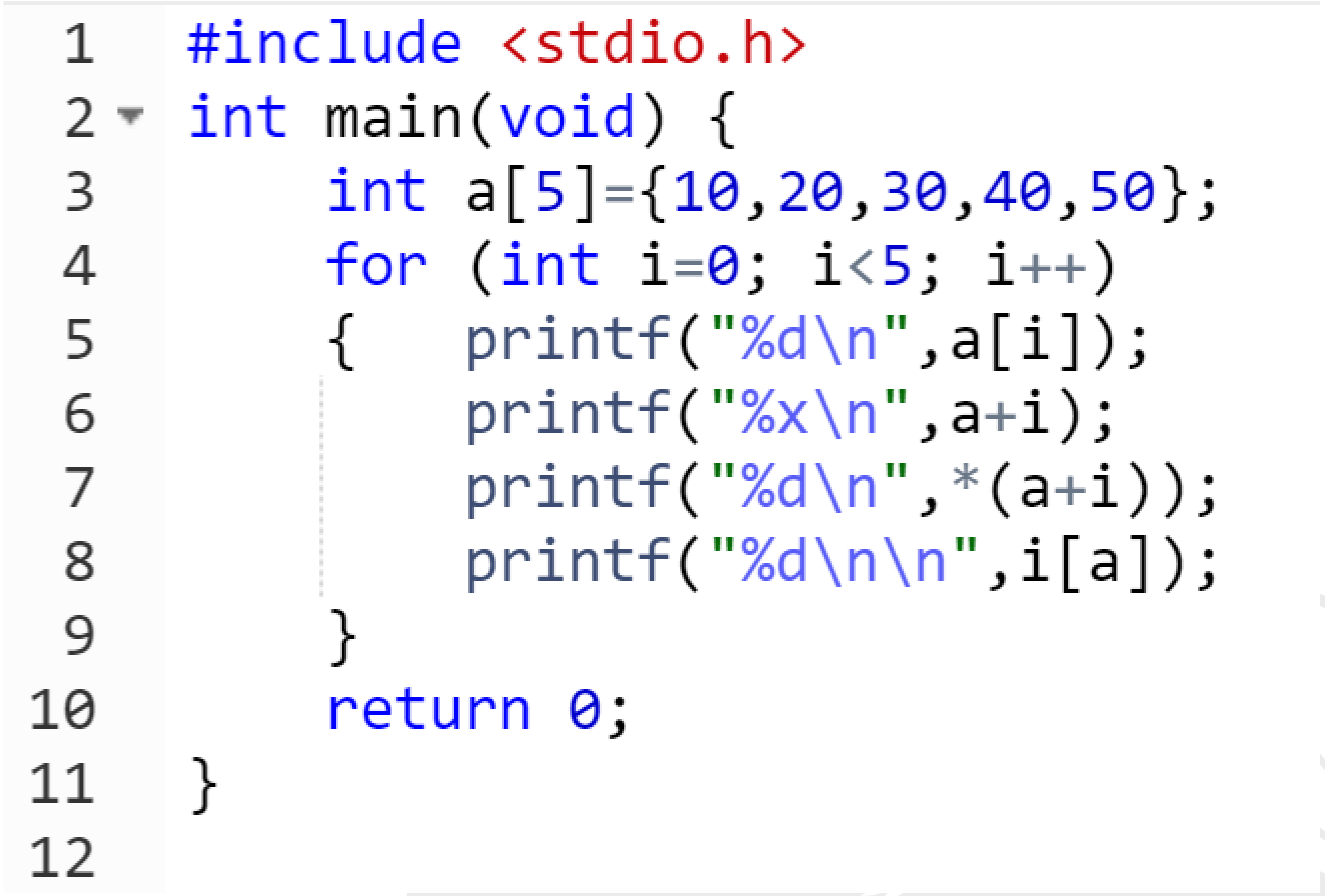
Runtime error #stdin #stdout 0s 5532KB

https://ideone.com/r7jKpj



```
~$ gcc DanglingPointer.c
~$ ./a.out
Segmentation fault (core dumped)
~$
```

Arrays and Pointers in C++



```
#include <stdio.h>
int main(void) {
	int a[5]={10,20,30,40,50};
	for (int i=0; i<5; i++)
	{	printf("%d\n",a[i]);
		printf("%x\n",a+i);
		printf("%d\n",*(a+i));
		printf("%d\n\n",i[a]);
	}
	return 0;
}
```

https://ideone.com/FNG35q

Thank You!