CS101C: Introduction to Programming (Using C)

Autumn 2025

Nikhil Hegde
Achyut Mani Tripathi

Week5: Recap of for, Arrays

So far..

- Library functions (printf, scanf)
- Data Types (int, float, double, char), constants, variables and their initialization using constants
- Storage for types
- Operators
- Control flow with if, if-else, else-if, switch, while, do-while, for

Today’s class (1/9/2025)

- Recap of for loop
- Arrays

Recap: `for` Statement - Syntax

```
for(expression1;expression2;expression3) {

statement1;

..

statementn;

}
```

- Expressions can be omitted. Semicolons must remain.
- Curly braces can be omitted when single statement present.
- Most commonly:
  - Expressions 1 and 3 are assignments
  - Expression2 is a relational expression

Recap: `for` Statement - meaning

```
for(expression1;expression2;expression3) {

statement1;

..

statementn;

}
```

1. Evaluate expression1.
2. Is expression2 true? (if expression2 not present, always true)
    a. If false execute the next statement after the `for` statement
3. Execute statements 1 to n.
4. Evaluate expression3.
5. Go to step 2.

Recap: Demo - while loop with comma operator

```c
#include <stdio.h>
int main()
{
    int i = 0, j = 0;
    while (i<5,j<10)
    {
        i++;
        j++;
    }
    printf("%d %d", i, j);
}
```

Recap: Demo - for loop

```
#include <stdio.h>
int main()
{
    short i;
    for (i = 1; i> 0; i++)
        printf("%d\n", i);
}
```

Recap: Demo - nested for loop

```
void main()
{
    int i = 0, j = 0;
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 1;)
        {
            break;
        }
        printf(" Bye \n");
    }
}
```

Recap: Demo - for loop with `double` type

```
#include <stdio.h>
void main()
{
    double k = 0;
    for (k = 0.0; k < 3.0; k++);
    printf("%lf", k);
}
```

Recap: Demo - infinite loop with `for`

```c
#include <stdio.h>
int main()
{
    for (5; 2; 2)
        printf("Hello\n");
    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    5;
   while(2) {
       printf("Hello\n");
       2;
   }
    return 0;
}
```

Challenge: What is the equivalent infinite loop with `while`?

Recap: `for and while equivalence`

```
#include <stdio.h>
int main()
{
    for (E1; E2; E3){
        stmt1;
        ..
        stmtn;
    }
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    E1;
    while(E2) {
        stmt1;
        ..
        stmtn;
        E3;
    }
    return 0;
}
```

Recap: for and while equivalence with continue stmt?

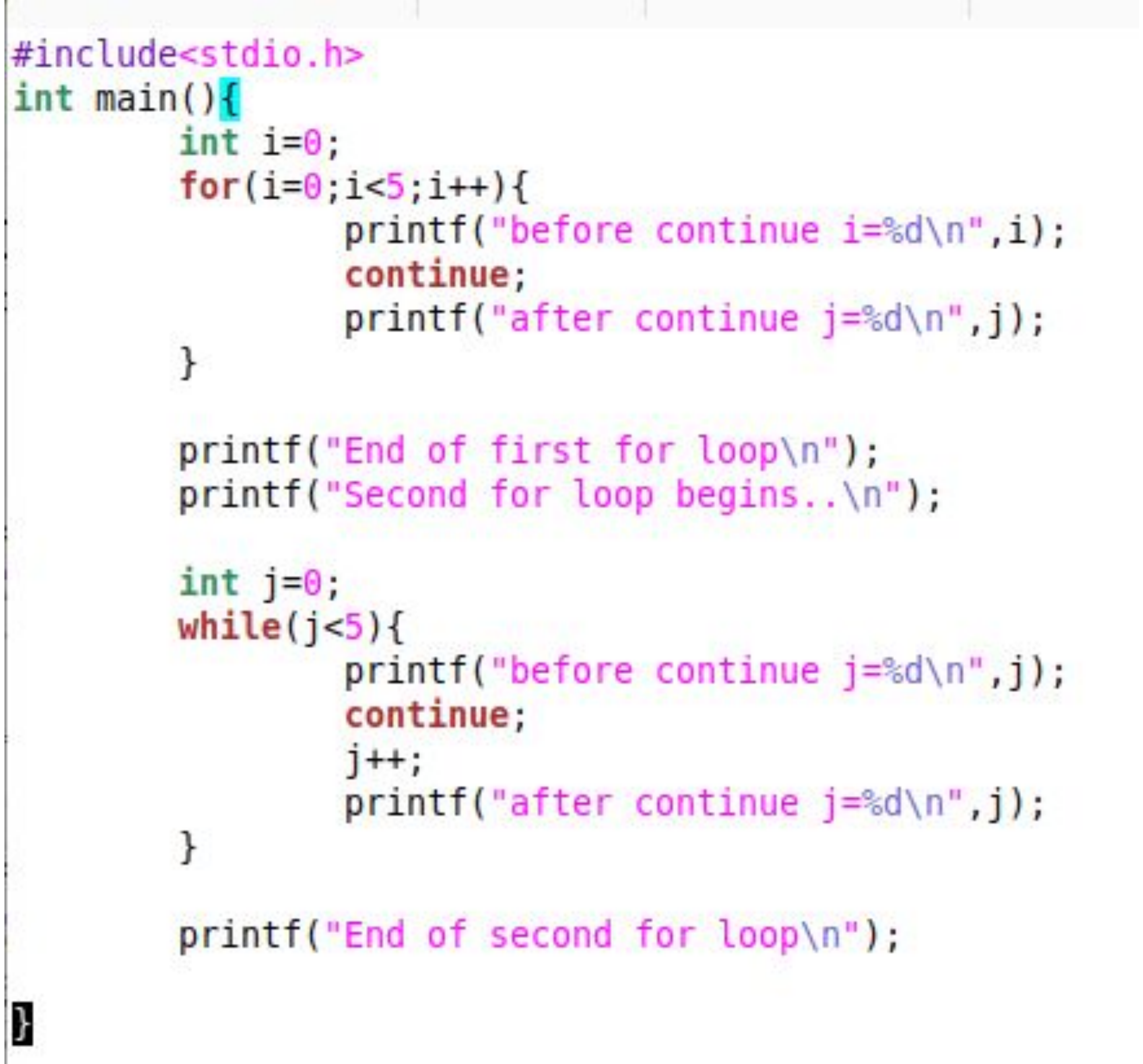
```
#include <stdio.h>
int main()
{
   for (E1; E2; E3){
      stmt1;
     continue;
     stmtn;
  }
 return 0;
}
```

Meaning is different!

Skip everything in loop body after continue. Execute E3 next

```
#include <stdio.h>
int main()
{
    E1;
   while(E2) {
      stmt1;
      continue;
      stmtn;
      E3;
   }
   return 0;
}
```

Skip everything in loop body after continue including E3!



```
#include<stdio.h>
int main(){
        int i=0;
        for(i=0;i<5;i++){
                printf("before continue i=%d\n",i);
                continue;
                printf("after continue j=%d\n",j);
        }

        printf("End of first for loop\n");
        printf("Second for loop begins..\n");

        int j=0;
        while(j<5){
                printf("before continue j=%d\n",j);
                continue;
                j++;
                printf("after continue j=%d\n",j);
        }

        printf("End of second for loop\n");

}
```

Agenda: Arrays

- Introduction to Arrays
- Definition
- Size and length of an array
- Modifying elements
- Demo programs:

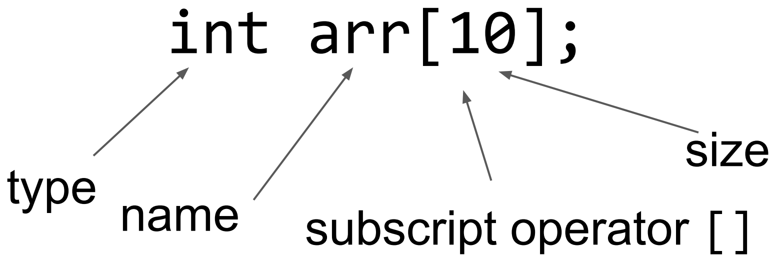
Today’s class (3/9/2025)

- Recap of Arrays
- More Arrays
    - Demo programs.

Arrays

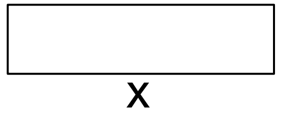
- Compound Data Type
- Allows you to store multiple elements with a single name.

E.g.



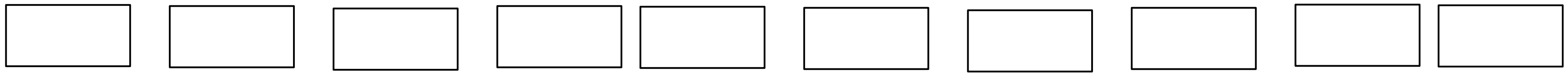
```
int arr[10];
```

- The size is fixed and must be a constant
- The subscript [ ] operator is needed to access elements.
- Recall `int x;` reserves a box in memory, which is named `x`.



Arrays

- `int arr[10];` reserves 10 boxes in memory.



These boxes are named as:

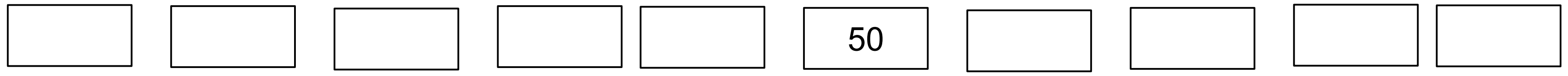
```
arr[0], arr[1], arr[2], …,arr[9]
```

- The array <u>index</u> always starts from zero in C

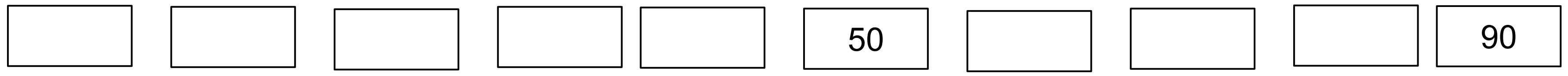
Arrays

- Now you know the name of a box. Reading a value and writing a value into the box is easy

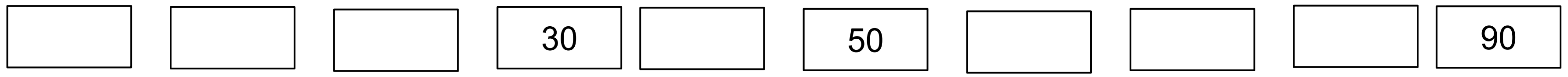
Write a value: arr[5]=50;



arr[9]=90;



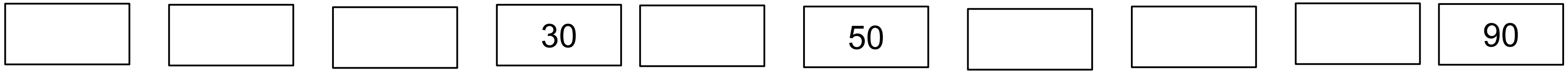
arr[3]=30;



Arrays

- Now you know the name of a box. Reading a value and writing a value into the box is easy

Read a value:



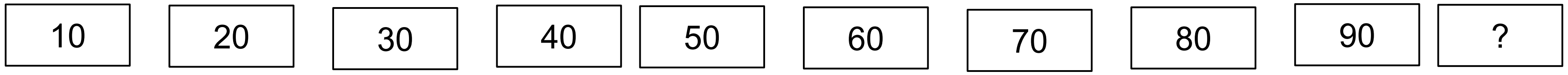
```
printf("%d", arr[3]);

int x=arr[5];
```

Arrays

- Initialize array:

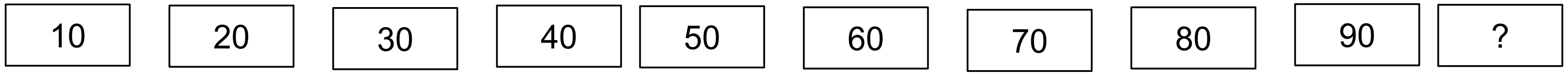
```
int arr[10]={10,20,30,40,50,60,70,80,90}
```



Arrays

- Initialize array:

```
int arr[10]={10,20,30,40,50,60,70,80,90}
```

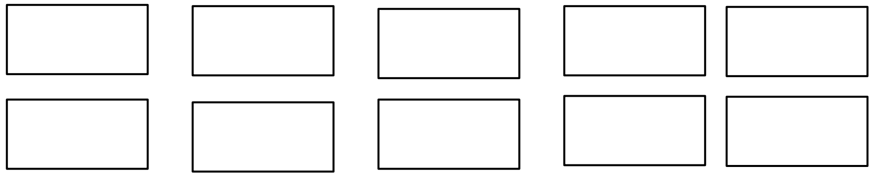


- Missing boxes are initialized to zero only if you use the initializer list.

Arrays

- Multi dimensional arrays:

```
int arr[2][5];
```



- Names of the boxes are:

```
arr[0][0], arr[0][1]...arr[0][4]

arr[1][0], arr[1][1]...arr[1][4]
```