ESC101: Introduction to Computing

Operators and Expressions

Practice Problem

- Write a program to read upto a maximum of 20 integers. Sum all the odd integers. Stop reading the sequence if you encounter a negative integer. Use break and continue

- Input: 2 3 8 1 5 -1
- Output: 9

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   int x,i=0,sum=0;
   for(i=0; i<20; i++)
   {
      scanf("%d",&x);
     if( _____)  //stop if negative
         _______;
     if( ________) //skip even
         ________;
      sum = sum+x;
   }
   printf("%d",sum);
   return 0;
}
```

Input: 4500

Output: 54

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   int x,i=0,sum=0;
   for(i=0; i<20; i++)
   {
      scanf("%d",&x);
     if( _____)  //stop if negative
         _______;
     if( 0 == x%2 ) //skip even
         continue;
      sum = sum+x;
   }
   printf("%d",sum);
   return 0;
}
```

Input: 4500

Output: 54

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   int x,i=0,sum=0;
   for(i=0; i<20; i++)
   {
      scanf("%d",&x);
     if( x <0 )  //stop if negative
         break;
     if( 0 == x%2 ) //skip even
         continue;
      sum = sum+x;
   }
   printf("%d",sum);
   return 0;
}
```

Input: 4500

Output: 54

Quick question

- Program to print all the alphabets

Output **ABCDEFGHIJKLMNOPQRSTUVWXYZ**

```
int main () {
    char ch;
    for (ch = ‘A’; ch <= ‘Z’; ch = ch +1) {
            printf(“%c”,ch);
    }
return 0;
}
```

Ternary operator ?:

- Select among values of two expressions based on a condition

```
condition ? true_expr : false_expr
```

- Both expressions must be of compatible type.
  - The expression is called ternary expression.

```
int abs;
int val;
scanf ("%d", val);
if (val < 0)
    abs = -val;
else
    abs = val;
printf("%d", abs);
```

```
int abs;
int val;
scanf ("%d", val);
abs = (val < 0) ? –val : val;
printf("%d", abs);
```

Condition

value if condition is True

value if condition is False

```
int val;
scanf ("%d", val);
printf("%d", (val < 0) ? –val : val);
```

Practice Problem

- Write a program to read in two numbers and use the ternary operator to compute the max of the two numbers and print it

- Input: 4 7
- Output: 7

Max using ternary operator

```
#include <stdio.h>
int main()
{
   int a,b,c;
   scanf("%d %d",&a,&b);
   c = (a > b)?a:b;
   printf(“%d\n”,c);
   return 0;
}
```

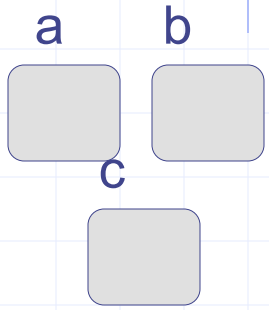
Comma as a separator

- C allows multiple variables of the same type to be defined as one statement, separated by commas.

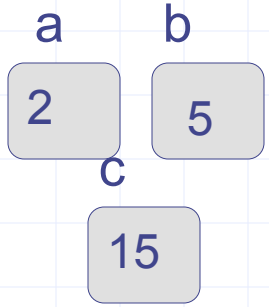
Examples (independent definitions)

```
int a, b, c;
int a = 2, b = 5, c=15;
float x = 3.59, y = 4.5;
int x = 5, float y = 10.0;
```

Defines three integer variables named a,b and c.



Defines three integer variables named a,b and c. Initializes a to 2, b to 5 and c to 15.



Compilation error!

Defines two float variables named x and y. Initializes x to 3.59 and y to 10.0.

Comma as a separator

- Also used in functions e.g., scanf and print for separating operands

```
scanf("%d%d", &a, &b);
printf("%d %d", a, b);
```

Comma as an operator

- Comma as an operator is a binary operator that takes two expressions as operands.

```
expr1 , expr2
```

- Think of , just like + or – or * or / or = or == etc.. Some examples,
  1. i+2, sum=sum-1;
  2. scanf("%d",&m), sum=0, i=0;
- Value associated with expr1 , expr2 :
- Evaluate expr1, discard its result and then evaluate expr2 and return its value (and type).

Comma Operator execution

- Commas are evaluated from *left to right*.

- The comma operator has the lowest precedence of all operators in C.

Comma Operator execution

- Commas are evaluated from *left to right*. That is,

  scanf("%d",&m), sum=0, i=0;

  is executed as

  (scanf("%d",&m), sum=0), i=0;

- The comma operator has the lowest precedence of all operators in C. So

  a=a+5, sum = sum + a

  is equivalent to

  (a=a+5), (sum = sum + a)

```
int a = 1; int sum = 5;
a=a+5,  sum = sum + a;
```

Examples

```
#include <stdio.h>
int main()
{
   int a;
   a = 3,3+4,3*5;
   printf("%d\n",a);
   return 0;
}
```

Output

3

```
#include <stdio.h>
int main()
{
   int a;
   a = (3,3+4,3*5);
   printf("%d\n",a);
   return 0;
}
```

Output

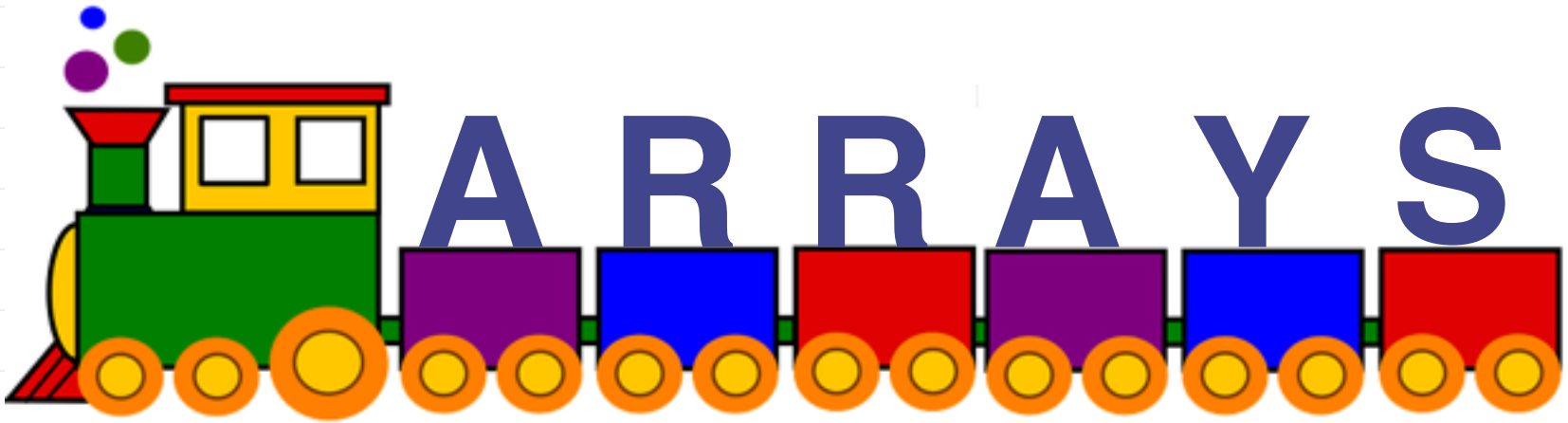
15

Comma as an operator

Expression	Evaluation
i = a, b;	(i=a),b;
i = (a, b);	Stores b into i
i = a, b, c;	Stores a into i discarding b and c
i = (a, b, c);	Stores c into i discarding a and b

```
for(i=0,j=i+1;i<=10;i++,j++)
```

ESC101: Introduction to Computing

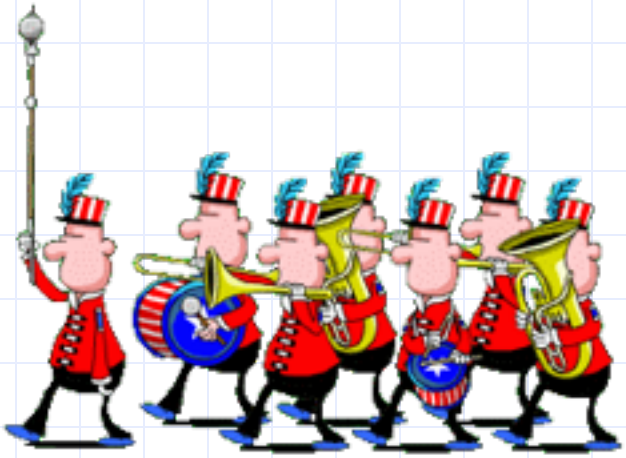


Defining arrays

Dictionary meaning of the word array

arr-ay: noun

1. a large and impressive grouping or organization of things: He couldn't dismiss the array of facts.
2. regular order or arrangement; series: an array of figures.



Arrays in C

An array in C is defined similar to defining a variable.

```
int a[5];
```

The square parenthesis [5] indicates that a is not a single integer but an array, that is a **consecutively allocated** group, of 5 integers.

It creates five integer boxes or variables

a[0] a[1] a[2] a[3] a[4]

The boxes are addressed as a[0], a[1], a[2], a[3] and a[4]. These are called the **elements** of the array.

```
include <stdio.h>
int main () {
      int i;
      int a[5];

       for (i=0; i < 5; i= i+1) {
            a[i] = i+1;

            printf("%d", a[i] );
        }
        return 0;
}
```

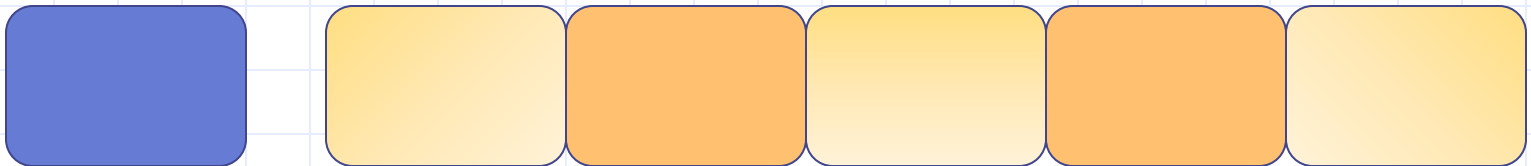
The program defines an integer variable called i and an integer array with name a of size 5

This is the notation used to address the elements of the array.

- The variable i is being used as an "index" for a.
- Similar to the math notation

a_i

i a[0] a[1] a[2] a[3] a[4]



```
include <stdio.h>
int main () {
    int a[5];
    int i;

    for (i=0; i < 5; i= i+1) {
        a[i] = i+1;
    }
    return 0;
}
```

Let us trace through the execution of the program.

Fact : Array elements are consecutively allocated in memory.

i 5

a[0]	a[1]	a[2]	a[3]	a[4]
1	2	3	4	5

Statement becomes a[0] =0+1;

Statement becomes a[1] =1+1;

Statement becomes a[2] =2+1;

Statement becomes a[3] = 3+1;

Statement becomes a[4] = 4+1;

One can define an array of float or an array of char, or array of any data type of C. For example

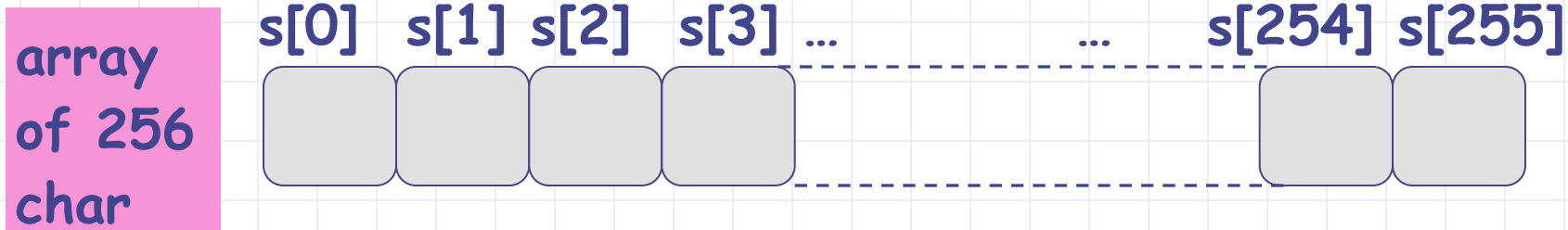
```
int main() {
        float num[100];

        char s[256];

 /* some code here */
}
```

This defines an array called num of 100 floating point numbers indexed from 0 to 99 and named num[0]… num[99]

This defines an array called s of 256 characters indexed from 0 to 255 and named s[0]…s[255].



Reading directly into array

Read N numbers from user directly into an array

```
#include <stdio.h>
int main() {
   int num[10];
   for (i=0; i<10; i=i+1) {
        scanf("%d", &num[i]);
   }
   return 0;
}
```

scanf can be used to read directly into array by treating an array element like any other variable of the same data type.

1. For integers, read as scanf("%d", &num[i]);
2. For reading elements of a char array s[], use scanf("%c", &s[j]).

In the previous slide, we had the statement:

What does &num[i] mean?

& is the "address-of" operator.

1. It can be applied to any defined variable.
2. It returns the location (i.e., address) of this variable in the program's memory.

[ ] is the array indexing operator, e.g, num[i].

```
scanf("%d", &num[i] );
```

&num[i] is made of two operators & and []. & num [i] gives the address of the array element num[i].

&num[i] is evaluated as &(num[i]).

& is applied to the result of applying the indexing operator [i] to num.

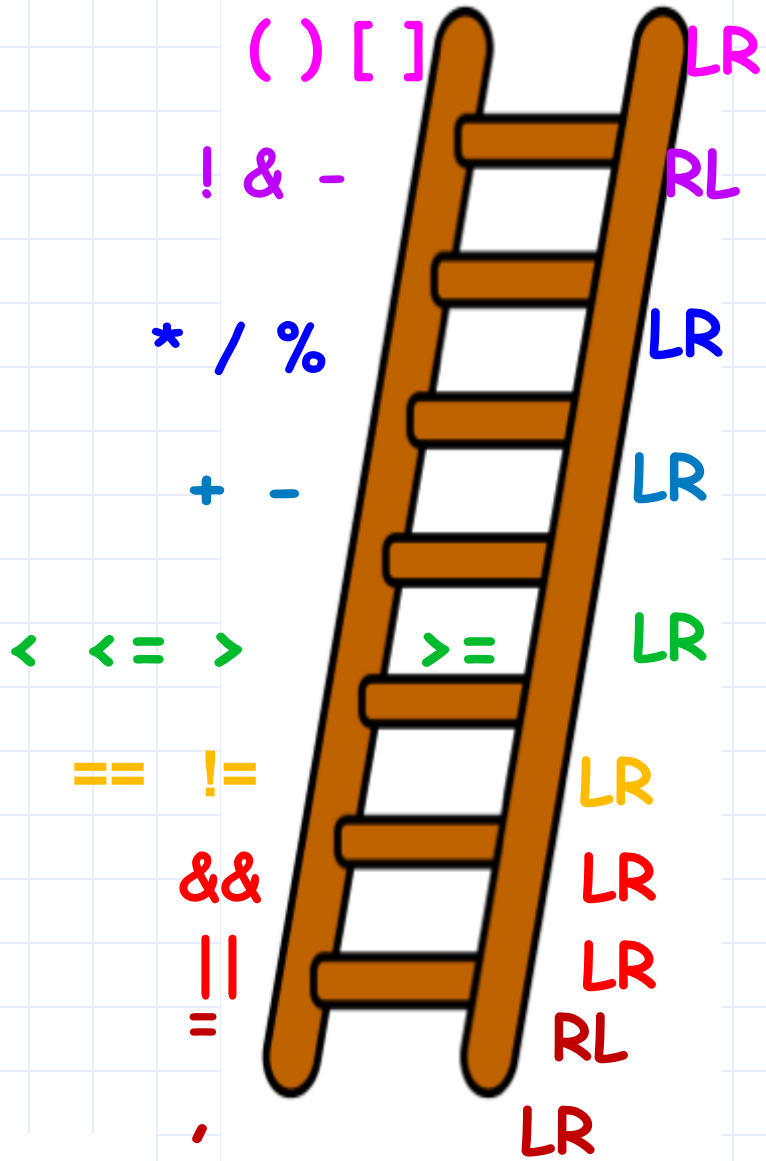
NOT as

(&num)[i] which would mean that first & is applied to num and [] operator is applied to &num

We have seen that &num[i] is evaluated by applying the indexing operator first and the address-of operator second.

More formally, the precedence of the operators in C reflects this.

1. The array indexing operator [] is given higher precedence than the address-of operator &.
2. So &num[i] is evaluated by applying the array operator first and the address-of operator next.



Legend LR: Left-to-Right associativity
RL: Right -to-Left associativity

Practice Problem

- Write a program to read in an array of 5 integers. Compute and print the running total of the integers.

- Input: 3 1 5 2 9
- Output: 3 4 9 11 20

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   int arr[5];
   //read input
   for(int i=0; i<5; i++)
    ________________

   //compute running total
   for(int i=1; i<5; i++)
    ________________

   //obtain output
   for(int i=0; i<5; i++)
      printf("%3d",arr[i]);

   printf("\n");
   return 0;

}
```

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   int arr[5];
   //read input
   for(int i=0; i<5; i++)
      scanf("%d",&arr[i]);

   //compute running total
   for(int i=1; i<5; i++)
    _______________

   //obtain output
   for(int i=0; i<5; i++)
      printf("%3d",arr[i]);

   printf("\n");
   return 0;

}
```

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   int arr[5];
   //read input
   for(int i=0; i<5; i++)
      scanf("%d",&arr[i]);

   //compute running total
   for(int i=1; i<5; i++)
      arr[i] = arr[i-1]+arr[i];

   //obtain output
   for(int i=0; i<5; i++)
      printf("%3d",arr[i]);

   printf("\n");
   return 0;

}
```

Practice Problem

- Write a program to read in an array of 5 integers and d a one element rotation of the input array.

- Input: 3 1 5 2 9
- Output: 1 5 2 9 3