ESC101 AM Class 1

Abhimanyu and Saranya

Overview

1. **Part 1** : Data I/O, Variables, Operators
2. **Part 2** : If-Else, Loops
3. **Part 3** : Debugging Code in Lab

Part 1
Data I/O, Variables and Operators

Structure of C Program

- **Comment Line**
- **Header**
- **Main Method declaration**
- **Body**
- **Return statement**

```
/* a code to introduce you to C
   programming */

// code begins here

#include <stdio.h>

int main()
{
   printf("Hello World");
   return 0;
}
```

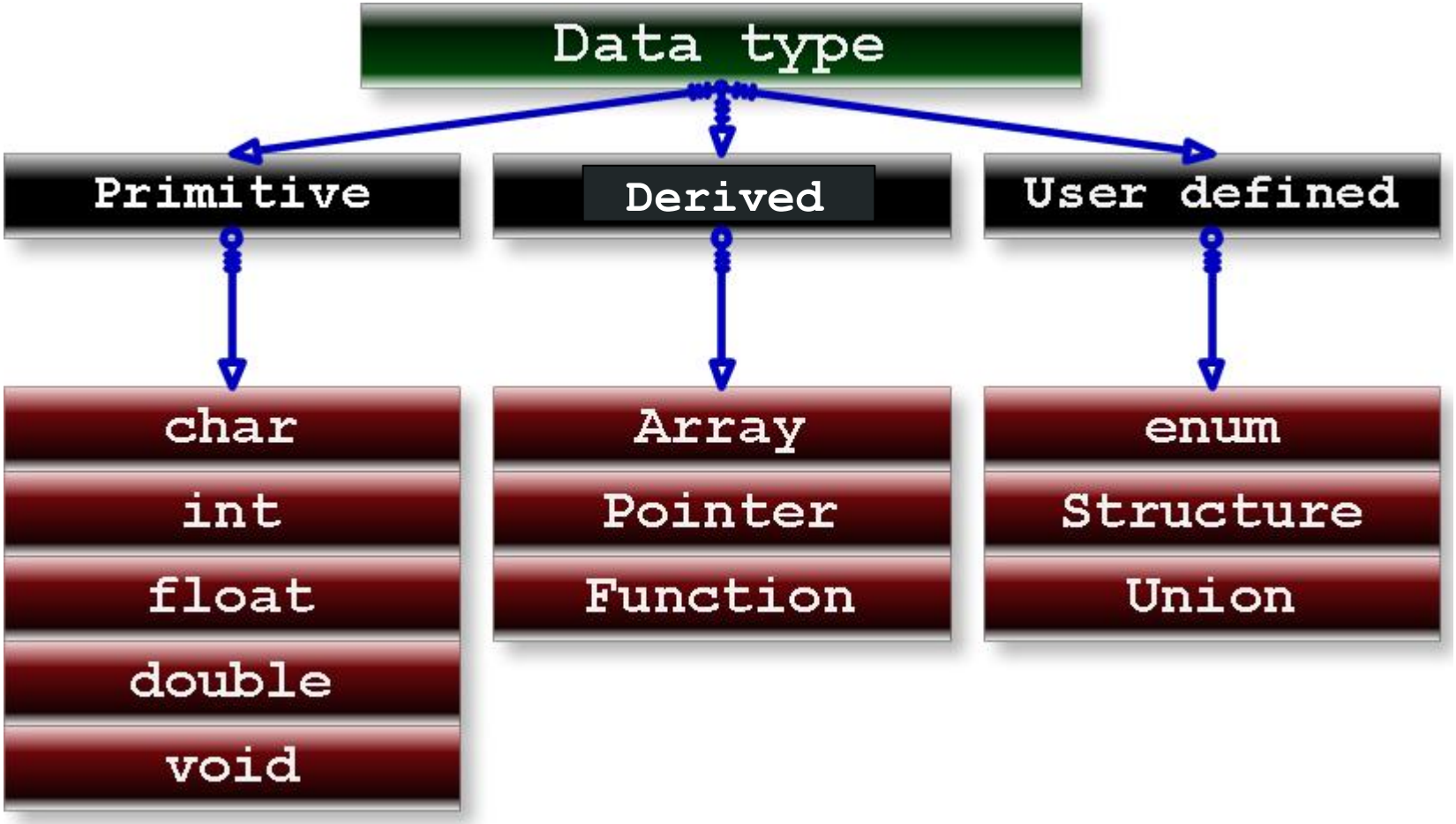
Variable Declaration and Initialization

Syntax:

datatype variablename;

datatype variablename = initial value;

```
#include <stdio.h>

int main()
{

    int x;
    int a = 2;

   // scanf(“%d”,&x);
   // printf("%d", a);

    return 0;
}
```

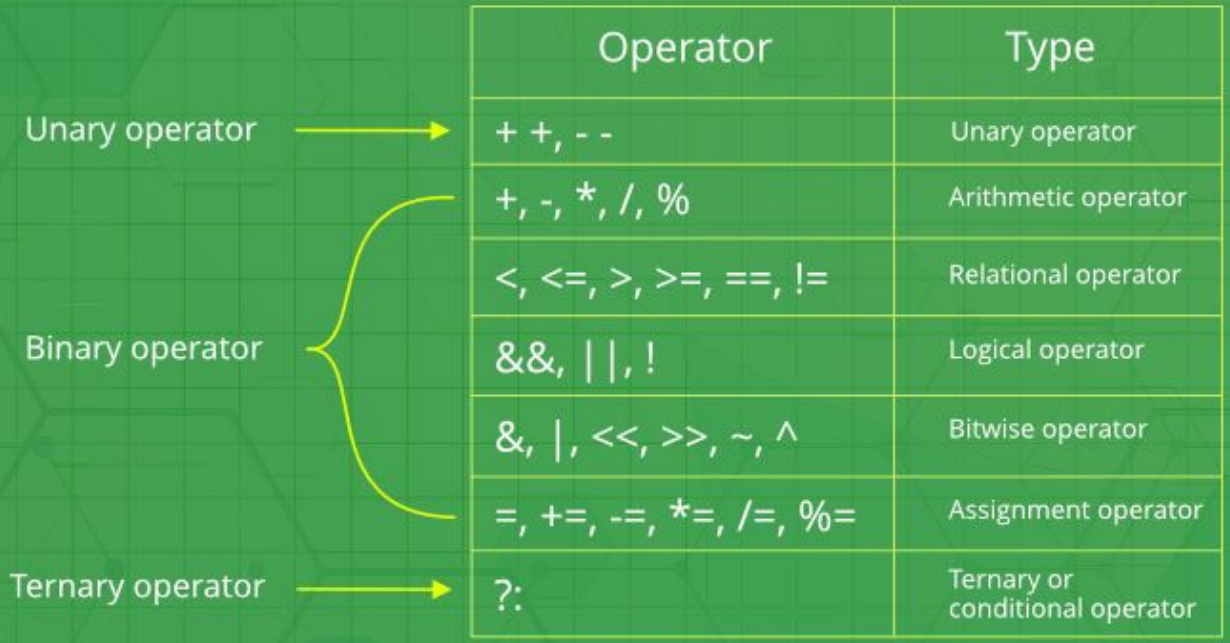

Data type
Primitive
Derived
User defined
char
int
float
double
void
Array
Pointer
Function
enum
Structure
Union

Printf and Scanf

%d - int
%f - float
%lf - double
%c - char

```
#include <stdio.h>

int main()
{

    int x;
    int a = 2;

   scanf(“%d”,&x);
   printf("%d", x);
   printf(“the value of a is %d”,a);

    return 0;
}
```

Operators in C



Arithmetic Operators

+, -, *, /, %

% modulus operator gives the remainder after integer division.

Example : 10%3 = 1

Assignment operator “=”
C = 1+5;

1. b= (1* a *a)/ 3
   Output = 1

2. float b;
   b = ⅓*a*a; printf(“%f”, b);
   Output = 0.0000

3. float b;
   b= (1* a *a)/ 3; printf(“%f”, b);
   Output = 1.0000

4. float b;
   b = (1*a*a)/3.0; printf(“%f”, b)
   Output = 1.3333

5. Homework for float a cases!

```
int a, b;
scanf(“%d”, &a);
b = ⅓*a*a;
printf(“%d”, b);
```

Input a -= 2
Output = 0
why?

1. int a = 2/3;
2. float a = 2/3;
3. int a = 2 /3.0;
4. float a = 2/3.0;
5. int a = 9/2;
6. float a = 9/2;

1. a will be 0 (no demotion/promotion)
2. a will be 0.0 (RHS is int with value 0, promoted to float with value 0.0)
3. a will be 0 (RHS is float with value 0.66, becomes int with value 0)
4. a will be 0.66 (RHS is float with value 0.66, no demotion/promotion)
5. a will be 4 (RHS is int with value 4, no demotion/promotion)
6. a will be 4.0 (RHS is int with value 4, becomes float with value 4.0)

Increment and decrement operator

Prefix : ++a, --a
Postfix : a++, a--

In the Pre-Increment, value is first incremented and then used inside the expression.

Whereas in the Post-Increment, value is first used inside the expression and then incremented.

Similarly with decrement operator.

Example

var1 is displayed
Then, var1 is increased to 6.

var2 is increased to 6
Then, it is displayed.

```
#include <stdio.h>
int main()
{
   int var1 = 5, var2 = 5;

   printf("%d\n", var1++);

   printf("%d\n", ++var2);

   return 0;
}
```

Relational Operators

>, < , >=, <=, ==, !=

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

A = 3 , B = 5

A == B

A+4 >= B+2

(A+B) > (A*2)

= is assignment operator.
== is a relational operator

Logical Operators

AND: &&
OR: ||
NOT: !

AND: (A&&B) If both the operands are non-zero, then the condition becomes true.

OR: (A||B) If any of the two operands is non-zero, then the condition becomes true.

NOT: !(A) It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false and vice versa.

Note:
if the first condition in && is False then the second condition is not checked

1. (5>=6 && 4>=2)

1. If c = 5 and d = 2 then, expression ((c==5) && (d>c)) equals to 0.

2. If c = 5 and d = 2 then, expression ((c==5) || (d>5)) equals to 1.

3. If c = 5 then, expression !(c==5) equals to 0.

2. x=4 y=5

a. ((++x >= y) && (++y ==6))
b. ((x++ >= y) && (++y ==6))
c. ((++x >= y) && (y++ ==6))

a. True, x=5, y=6
b. False, x=5, y=5
c. False, x=5, y=6

Precedence and Associativity

1. **int a=20, b=10, c=15, d=5;**
   e = (a + b) * c / d ;
   // (30 * 15) / 5

Associativity is used when two operators of same precedence appear in an expression.

2. **int a=2, b;**
   b = a++ + a/2 * 5 > 6 && a == 3;

   a= 3, b = 1

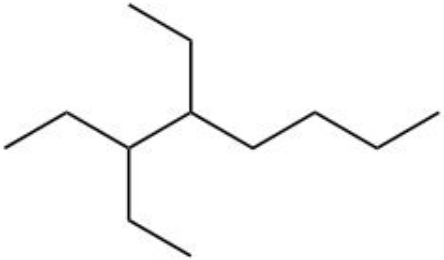
Level	Operators	Description	Associativity
15	() [] -> . ++ --	Function Call Array Subscript Member Selectors Postfix Increment/Decrement	Left to Right
14	++ -- + - ! ~ (type) * & sizeof	Prefix Increment / Decrement Unary plus / minus Logical negation / bitwise complement Casting Dereferencing Address of Find size in bytes	Right to Left
13	* / %	Multiplication Division Modulo	Left to Right
12	+ -	Addition / Subtraction	Left to Right
11	>> <<	Bitwise Right Shift Bitwise Left Shift	Left to Right
10	< <= > >=	Relational Less Than / Less than Equal To Relational Greater / Greater than Equal To	Left to Right
9	== !=	Equality Inequality	Left to Right
8	&	Bitwise AND	Left to Right
7	^	Bitwise XOR	Left to Right
6	|	Bitwise OR	Left to Right
5	&&	Logical AND	Left to Right
4	||	Logical OR	Left to Right
3	?:	Conditional Operator	Right to Left
2	= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operators	Right to Left
1	,	Comma Operator	Left to Right

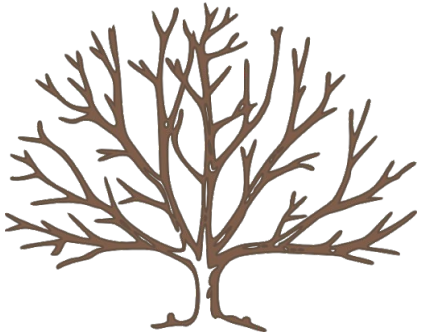
Part 2

Control Flow

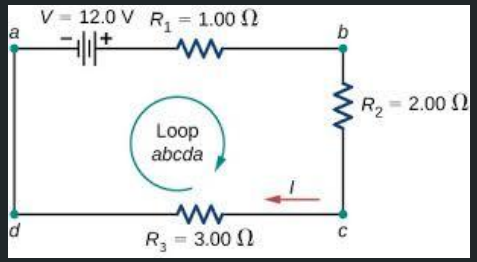
Branching and looping

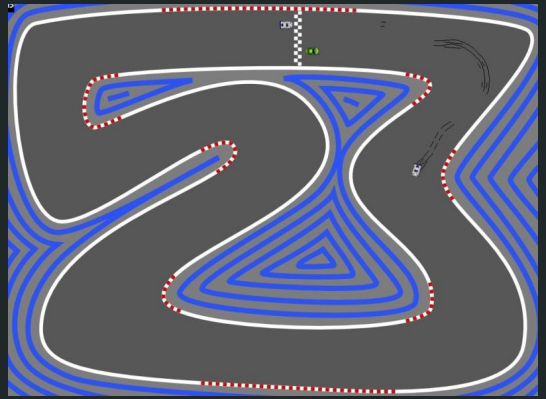
Branching





Looping





Branching

1

```
if (condition 1){
    action 1
    action 2
}
```

2

```
if (condition 1){
    action 1
}else{
    action 2
}
```

3

```
if (condition 1){
    action 1
}else if (condition 2){
    action 2
}else{
    action 3
}
```

Branching

4

```
if (condition 1){
    action 1
    if (condition 2){
        action 2
        action 3
    }
    action 4
}
```

5 Switch statement

6 Ternary operator
condition1?actions1:action2

Common Errors

No bracket after if

If no bracket is applied after an if statement,

then only the statement immediately after if is considered in the scope of the if block

```
if (condition 1)
    action 1;
    action 2;
```

If condition 1 is False, then which of the actions will be implemented-

(A) Action 1 and 2 both
(B) Action 2 but not Action 1
(C) Action 1 but not Action 2
(D) Neither Action 1 nor Action 2

Common Errors

Unmatched else statement

You can have an if without an else

But NOT an else without an if

```
if (condition1){
    action 1;
}else{
    action 2;
}else{
    action 3;
}
```

Common Errors

Unmatched if else

An else always matches to the closest unmatched if statement

```
if (condition 1)
    action 1;
if (condition 2)
    action 2;
else
    action 3;
```

..unless forced otherwise by {....}

```
if (condition 1) {
    action 1;
if (condition 2)
    action 2;
} else
    action 3;
```

If condition 1 is False and condition 2 is True, then which of the actions will be implemented -

(A) Action 2 and 3 both

(B) Action 2 only

(C) Action 3 only

(D) All- Action 1, 2 and 3

Looping

Print all natural numbers till n

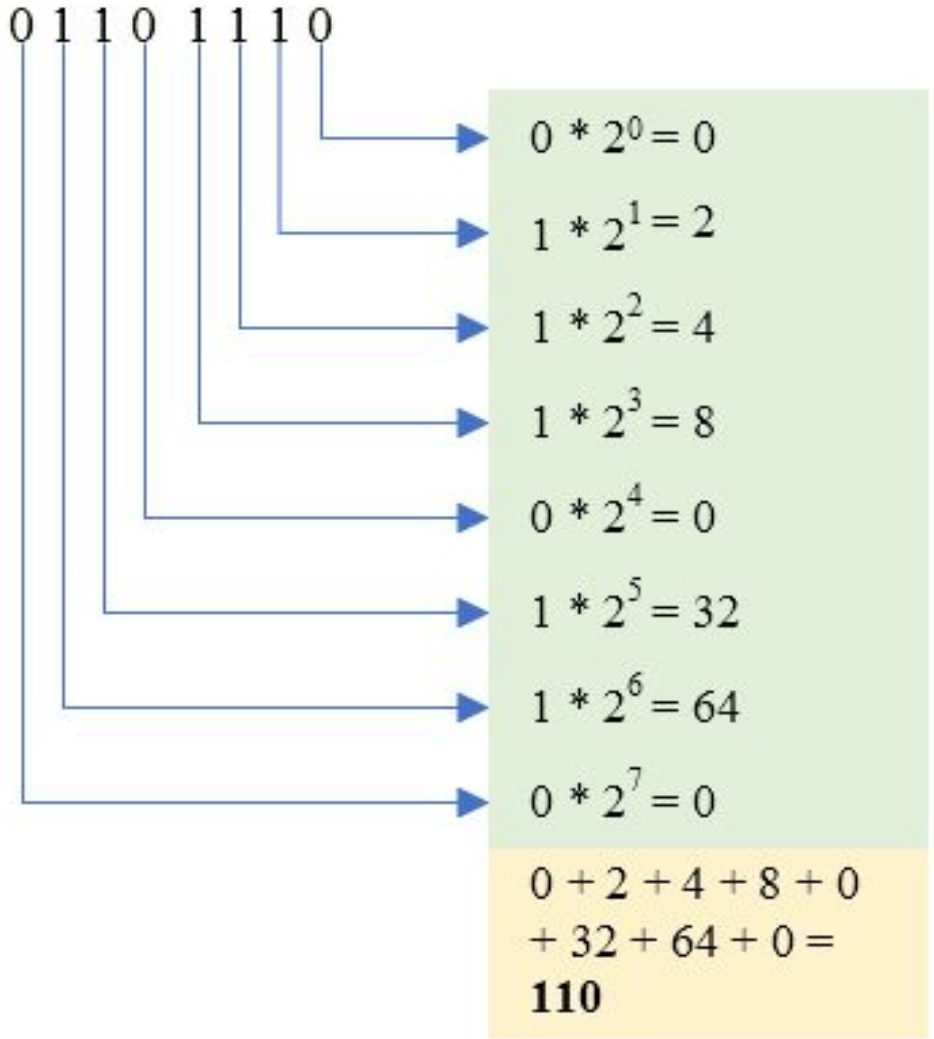
```
int x=1;
while(x<=n){
    printf("%d ",x);
    x++;
}
```

Solving a Loop Problem

1. What do you want to repeat each time?
    a. Print statement
2. What value do you want to change each time?
    a. Variable x by 1
3. When do you want to stop looping?
    a. When variable x reaches n

Convert Binary numbers to Decimal number

Looping Intuition: Repeating the Process for each digit



1. **What do you want to repeat each time?**

   Repeating the multiplication

   Digit * 2^(digit index)

```
main(){
    int bin;
    scanf("%d",&bin)
    int decimal;

    int dig_index = 0;
    decimal = 0;

    while(      ){
        int digit = bin%10;
        decimal += digit * pow(2,dig_index)

    }

}
```

2. What value do you want to change each time?

Increase digit index by 1

bin = bin/10

```
main(){
    int bin;
    scanf("%d",&bin)
    int decimal;


    int dig_index = 0;
    decimal = 0;

    while(      ){
        int digit = bin%10;
        decimal += digit * pow(2,dig_index)
        dig_index ++;
        bin = bin/10


    }


}
```

3. When do you want to stop looping?

When all digits are exhausted

I.e. bin == 0

```
main(){
    int bin;
    scanf("%d",&bin)
    int decimal;


    int dig_index = 0;
    decimal = 0;

    while(bin!=0){
        int digit = bin%10;
        decimal += digit * pow(2,dig_index)
        dig_index ++;
        bin = bin/10


    }


}
```

Part 3

Tips and Tricks

....to get through labs 😫😫

Debugging Your Code

Case 1: Visible test cases are passing

Not sure where the error is?

1. Print the temporary variables, run and check if the variable is updating as expected
2. Test each block separately by commenting out the rest of the code
3. If visible test cases do not pass, simulate mentally/on paper the code for the particular input

Debugging Your Code

Case 1: Visible test cases are not passing

But my output is correct?

then look for formatting mistakes

(extra space/ extra line/ etc.)

Debugging Your Code

Case 2: Hidden test cases not passing

Look for border values

Overflow errors- try and change datatype

Thank You!

Doubts?

Abhimanyu Sethia
+918989624974
fb.com/abhimanyusethia12
Saranya Satheesh
+917356570705
fb.com/chaaruusaranya