C Programming Advanced

Ashish Nanda

Ashish.Nanda@uts.edu.au

Data Types: Arrays

A variable can be declared in two ways:

- **int a; char a; float a;**
  - Here a will store a single value

- **int a[x]; char a[x]; float a[x];**
  - Here a can store multiple values accessible using the index number **x**.
  - Each index number is associated with a unique memory address

Note: **char a[x]** is referred to as the string datatype as it can store multiple characters (like a word) at once. It does so by splitting the word and storing single character at each index value.

int a char a float a	
Variable	**Memory Address**
a	(0xaa1122)
int a[x] char a[x] float a[x]	
Variable	**Memory Address**
a[0]	(0xa71522)
a[1]	(0xad1142)
a[2]	(0xab3162)
a[3]	(0xba1472)
a[4]	(0xbb1822)
a[5]	(0xab1129)

if & if else statement

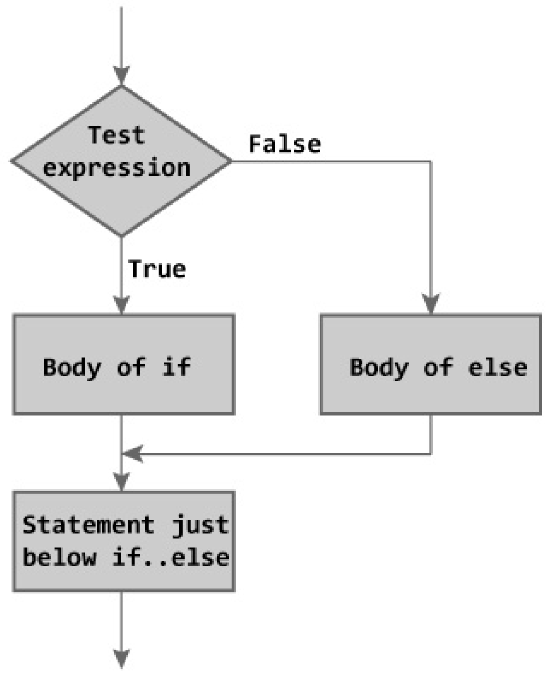


Figure: Flowchart of if...else Statement

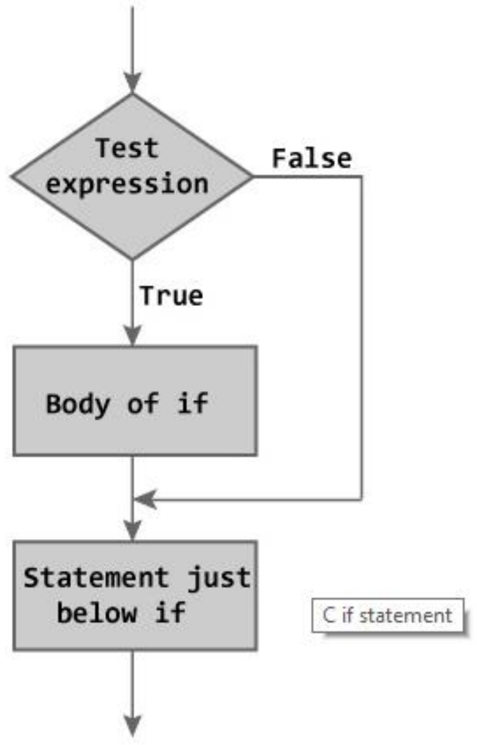
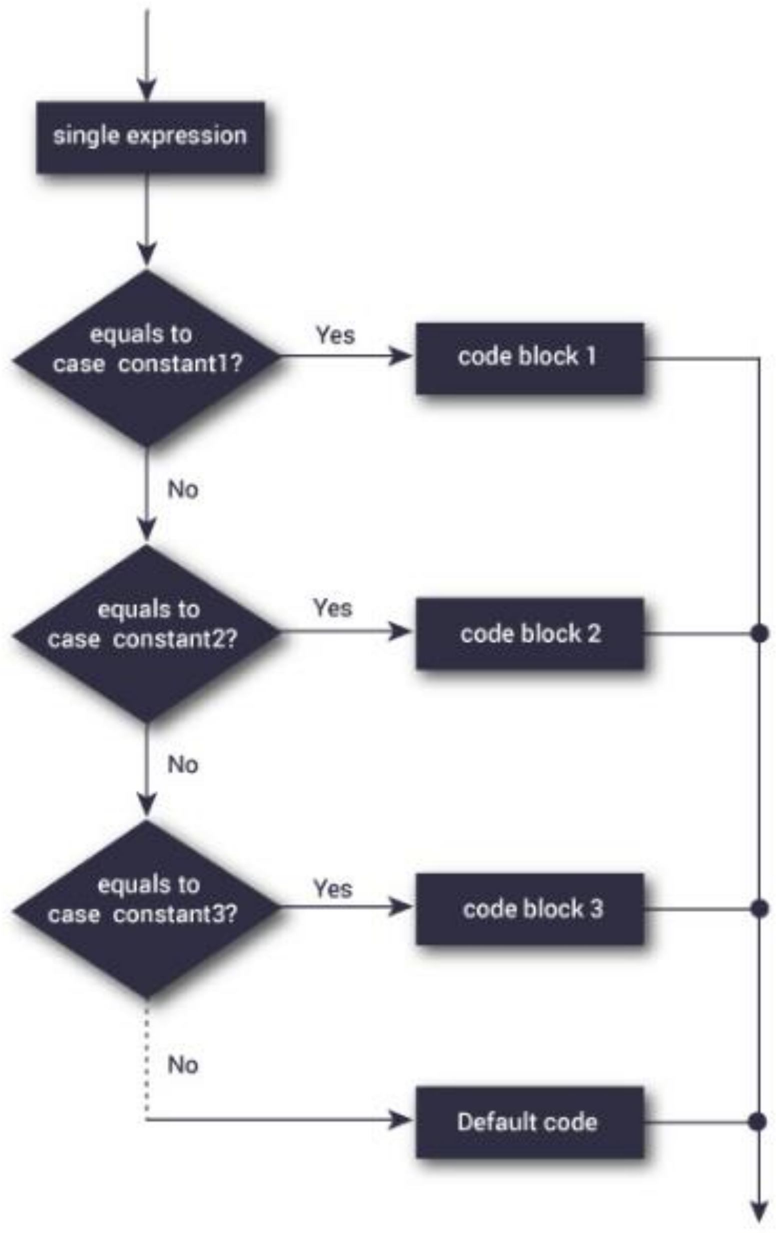


Figure: Flowchart of if Statement

Lets try a program

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int age;
    printf( "Please enter your age" );
    scanf( "%d", &age );
    if ( age < 100 )
    {
            printf ("You are pretty young!\n" );
    }
    else if ( age == 100 )
    {
            printf( "You are old\n" );
    }
    else
    {
            printf( "You are really old\n" );
    }
    return 0;
}
```

Switch Case


single expression
equals to case constant1?
Yes
code block 1
No
equals to case constant2?
Yes
code block 2
No
equals to case constant3?
Yes
code block 3
No
Default code

Lets try a program

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int inta, intb, input;
    printf("Please enter two numbers \n");
    scanf("%d %d", &inta, &intb);
    printf("Please make a selection by entiring the number: \n ");
    printf("Select 5 for Addition: \n ");
    printf("Select 6 for Subtraction: \n ");
    printf("Select 7 for Multiplication: \n ");
    scanf("%d", &input);
    switch ( input )
    {
    case 5:
                    printf("The answer is: %d", (inta + intb)) ;
                    break;
    case 6:
                    printf("The answer is: %d", (inta - intb)) ;
                    break;
    case 7:
                    printf("The answer is: %d", (inta * intb)) ;
                    break;
    default:
                    printf( "Bad input, quitting!\n" );
                    break;
  }
  return 0;
}
```

FOR LOOP

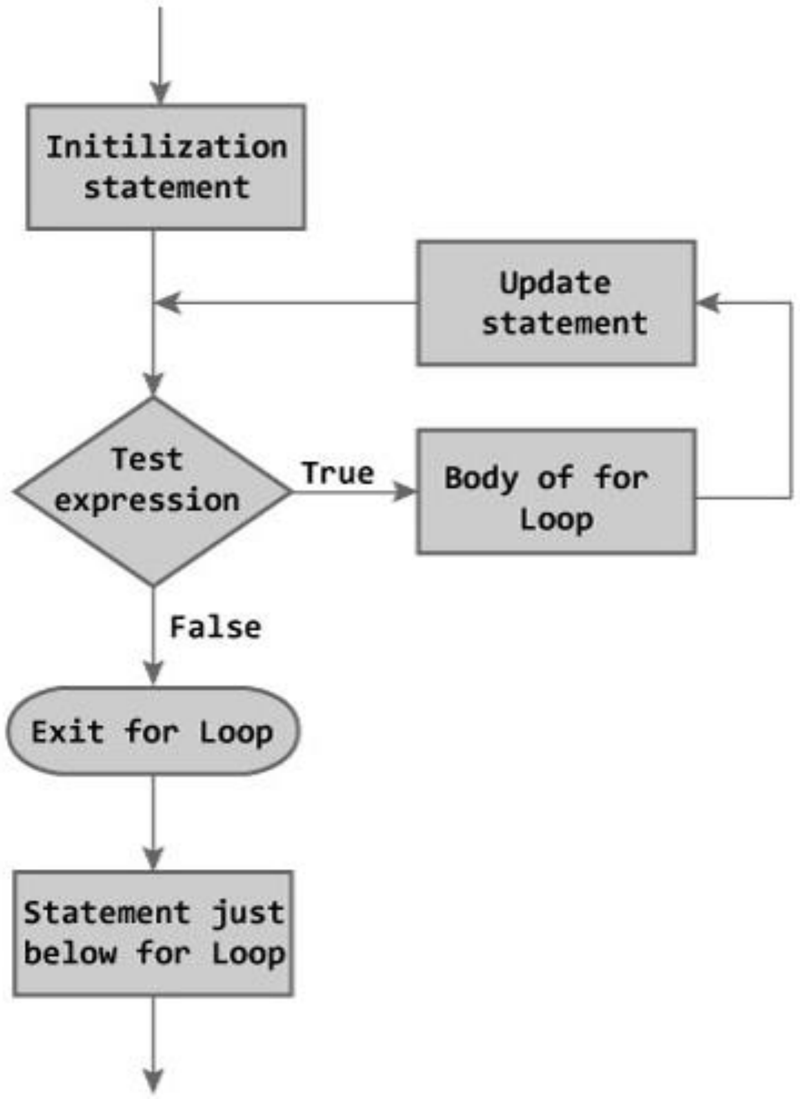


Figure: Flowchart of for Loop

Lets try a program

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int x;
   for (x = 0; x < 10; x++)
   {
               printf("%d\n", x );
   }
   return 0;
}
```

WHILE LOOP

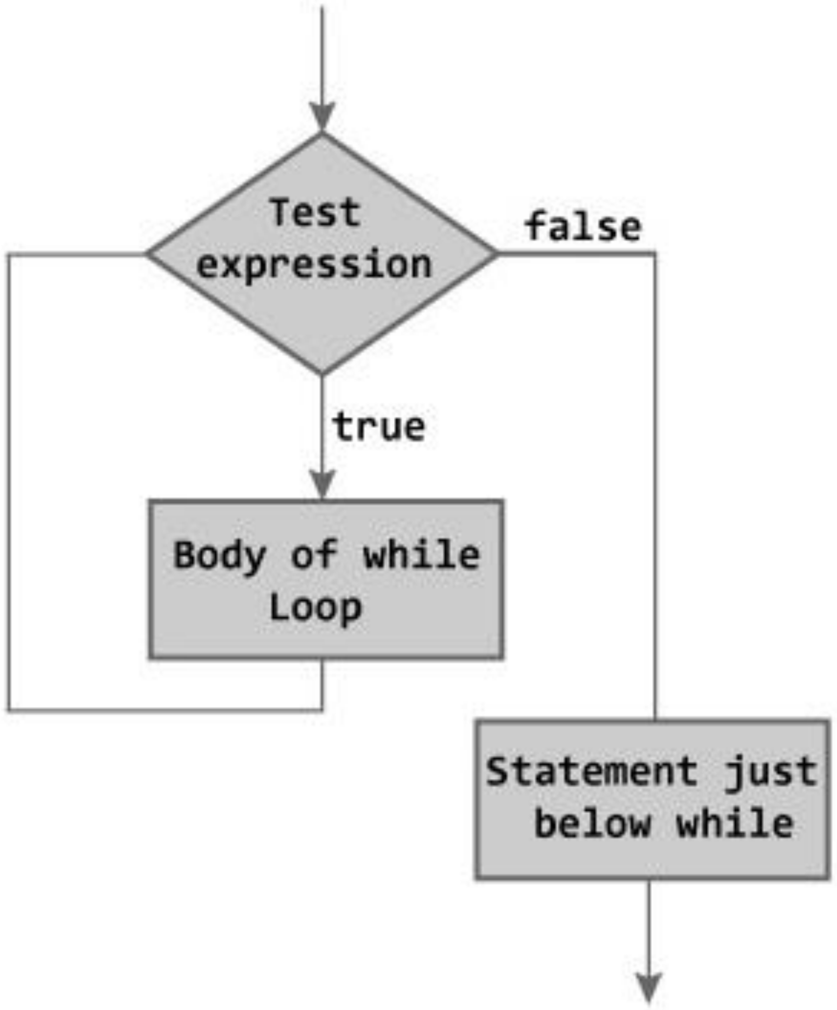


Figure: Flowchart of while Loop

Lets try a program

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int x = 0;
   while (x < 10)
   {
               printf("%d\n", x );
               x++;
   }
   return 0;
}
```

DO - WHILE LOOP

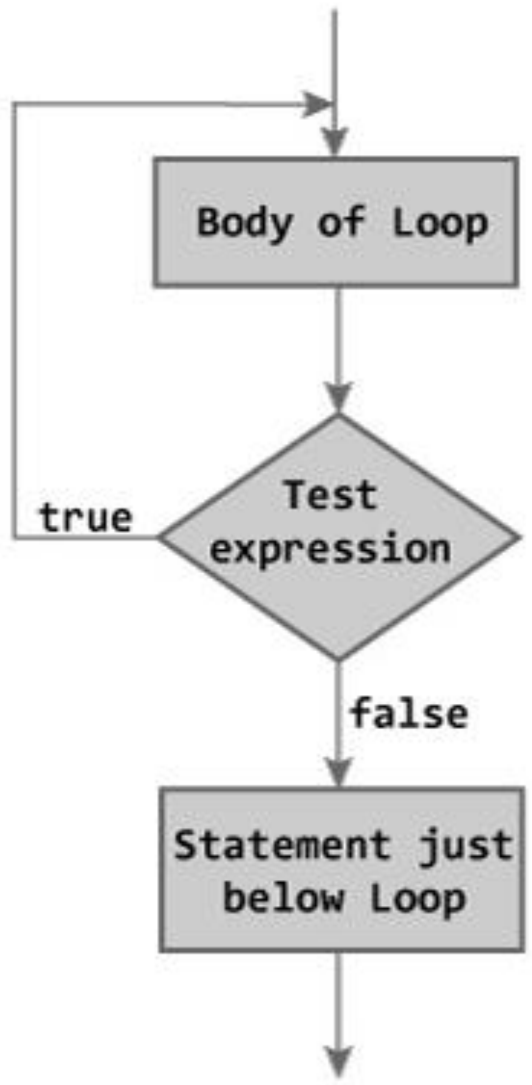


Figure: Flowchart of do...while Loop

Lets try a program

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int x = 0;
   do
   {
               printf("%d\n", x );
               x++;
   } while (x < 10) ;
   return 0;
}
```

Functions

- Functions are an important part of programming
- It can be used to define a set of actions
- The function can be called into the main program multiple times
- The function can also call itself
- A function must be defined / declared before the main program. This is because the compiler follows a top to bottom approach.

Lets try a program

```
#include <stdio.h>
#include <stdlib.h>

int mult (int x, int y);
int main()
{
   int num1, num2, cal;
   printf("Please input two numbers to be multiplied: ");
   scanf("%d %d", &num1, &num2);
   cal = mult(num1, num2);
   printf("The product of your two numbers is %d\n", cal);
   return 0;
}

int mult (int x, int y)
{
   int calc;
   calc = x * y;
   return calc;
}
```

Pointers

- Pointer variables are variables that store(point to) memory addresses.
- Pointer Declaration:
  - **int x, y = 5;**
  - **int *ptr;**
  - ***ptr** is a POINTER to an integer variable*/

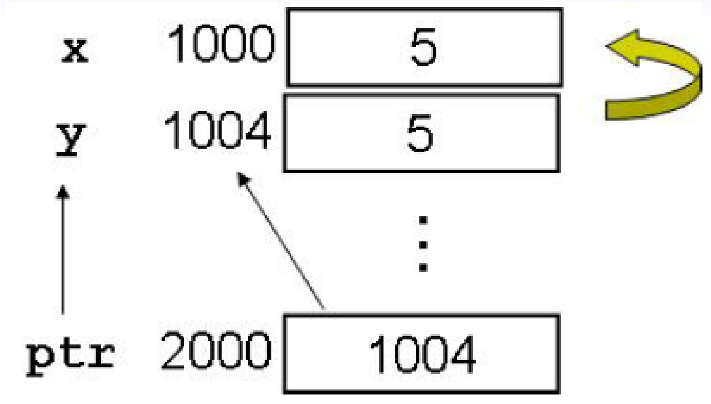
- Reference operator **&** when used before a variable will refer to its memory location
  - **ptr = &y;**
  - This will assign **ptr** to the MEMORY ADDRESS of **y**

- Dereference operator *****
  - **x = *ptr;**
  - This will assign **x** to the **int** that is pointed to by **ptr**, this is the same as writing **x = y**

Pointer Examples

Pointer Example 1

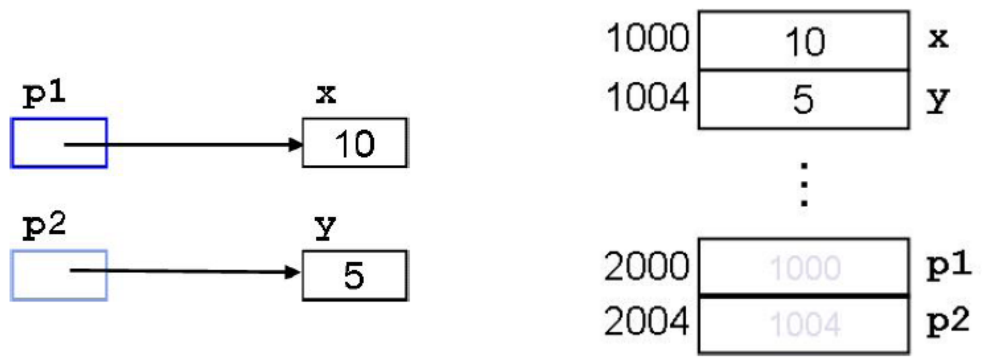
```
int x;
int y = 5;
int *ptr;

ptr = &y;

x = *ptr;
```



Pointer Examples

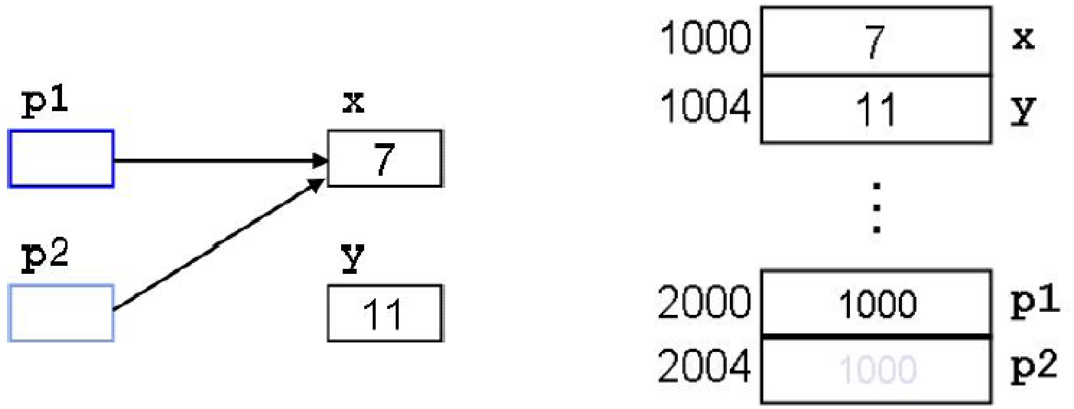
Pointer Example 2

```
int x = 10, y = 5;
int *p1, *p2;
p1 = &x;
p2 = &y;
```



Pointer Example 2

```
p2 = p1;   // Not the same as *p2 = *p1
```



Lets try a program

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int* pc;
   int c;
   c = 22;
   printf("Address of c:%p \n",&c);
   printf("Value of c:%d \n\n",c);
   pc = &c;
   printf("Address of pointer pc:%p \n",pc);
   printf("Content of pointer pc:%d \n\n",*pc);
   c=11;
   printf("Address of pointer pc:%p \n",pc);
   printf("Content of pointer pc:%d \n\n",*pc);
   *pc=2;
   printf("Address of c:%p \n",&c);
   printf("Value of c:%d \n\n",c);
   return 0;
}
```

File I/O

- Useful for reading and writing to external files.
- Uses different modes:
  - r - open for reading (file should exist)
  - w - open for writing (file need not exist)
  - a - open for appending (file need not exist)
  - r+ - open for reading and writing(start at beginning)
  - w+ - open for reading and writing (overwrite file)
  - a+ - open for reading and writing (append if file exists)

Lets try a program

Before writing the program, We need to create a file and add some data.

Use the following command in terminal to create the file

gedit in.list

Once the file opens up add the following data to the file:

foo 70
bar 98
biz 100

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *ifp, *ofp;
    char username[9];
    int score;
    ifp = fopen("in.list", "r");
    if (ifp == NULL) {
                    printf("Can't open input file in.list!\n");
                    exit(0);
    }
    ofp = fopen("out.list", "w");
    if (ofp == NULL) {
                    printf("Can't open output file out.list!\n");
                    exit(0);
    }
    while (fscanf(ifp, "%s %d", username, &score) == 2) {
                    fprintf(ofp, "%s %d\n", username, score+10);
    }
    fclose(ifp);
    fclose(ofp);
    return 0;
}
```