# Programming with C and C++

*CSC-101 (Lecture 29 and 30)*

**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/*

# Call by Reference

https://ideone.com/qNDMqY

```
1  #include <iostream>
2  using namespace std;
3
4  void swapVariables(float* var1, float* var2)
5      { float temp;
6          temp = *var1;
7          *var1 = *var2;
8          *var2 = temp;
9          cout<<*var1<<" "<<*var2<<endl;
10     }
11
12 int main( )
13     { float float1 = 3.5, float2 = 5.6;
14         swapVariables(&float1, &float2);
15         cout<<float1<<" "<<float2<<endl;
16         return 0;
17     }
18
```

stdout

```
5.6 3.5
5.6 3.5
```

# Call by value and reference

```
1  #include <iostream>
2  using namespace std;
3
4  void g(int x, int& y);
5
6  int main() {
7      int a,b;
8      a=20;
9      b=15;
10     g(a,b);
11     cout<<a<<" "<<b<<endl;
12     g(5*a-2,b);
13     cout<<a<<" "<<b<<endl;
14
15     return 0;
16 }
17
```

https://ideone.com/QLlbRB

```
18  void g(int x, int& y)
19      { x=50;
20        y=80;
21       }
22
```

stdout

```
20 80
20 80
```

# Call by value and reference

```
1   #include <iostream>
2   using namespace std;
3
4   void g(int x, int& y);
5
6   int main() {
7       int a,b;
8       a=20;
9       b=15;
10      g(a,b);
11      cout<<a<<" "<<b<<endl;
12      g(5*a-2,b);
13      cout<<a<<" "<<b<<endl;
14      g(a,5*b-3);
15      cout<<a<<" "<<b<<endl;
16
17      return 0;
18  }
19
```

https://ideone.com/MqDNQa

```
20  void g(int x, int& y)
21      { x=50;
22        y=80;
23       }
24
```

compilation info

```
prog.cpp: In function ‘int main()’:
prog.cpp:14:9: error: cannot bind non-const lvalue reference of type ‘int&’ to an rval
ue of type ‘int’
  g(a,5*b-3);
      ~~~^~
prog.cpp:4:6: note:   initializing argument 2 of ‘void g(int, int&)’
 void g(int x, int& y);
      ^
```

# Static Array

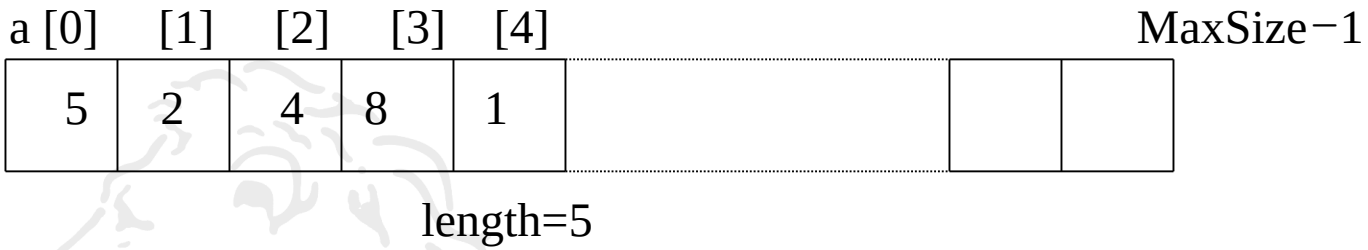
```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      // your code goes here
6      int a[3];
7      for (int i=0; i<3; i++)
8          cin>>a[i];
9      for (int i=0; i<3; i++)
10         cout<<a[i];
11     delete [] a;
12     return 0;
13 }
14
```

https://ideone.com/9I0q9q

Runtime error

# One Dimensional Arrays

- Contiguous: `int a[10];`




- Linked (Using Pointers)

```
int* a;
a=new int[10];
```

```
int* a;
int n=10;
while (n>0)
 { a=new int[n];
for (int i=0; i<n;i++)
 cin>>a[i];
sort(a,n); //som function
n--;
delete [ ] a;
}
```

# Global variable in C++

</> source code

https://ideone.com/v2zqbH

```
1  #include <iostream>
2  using namespace std;
3
4  int a[1];
5  int main() {
6      // your code goes here
7      cout<<"Array concept is so funny in C++"<<endl;
8      for (int i=10; i<20;i++)
9       a[i]=i+1;
10
11      for (int i=10; i<20;i++)
12       cout<<a[i]<<endl;
13     return 0;
14 }
```

# Global variable in C++

Success #stdin #stdout 0s 5420KB

```
Array concept is so funny in C++
11
12
13
14
15
16
17
18
19
20
```

# Dynamic Array

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // your code goes here
6       int* a;
7       a=new int[10];
8       for (int i=0; i<10;i++)
9        a[i]=i+1;
10       cout<<"Before Deallocation"<<endl;
11
12       for (int i=0; i<10;i++)
13        cout<<a[i]<<endl;
14
15        delete [] a;
```

# Dynamic Array

```
16
17          cout<<"After Deallocation"<<endl;
18
19          for (int i=0; i<10;i++)
20          cout<<a[i]<<endl;
21
22      return 0;
23  }
```

https://ideone.com/Snmsqp

# Output

Success #stdin #stdout 0.01s 5500KB

| Before Deallocation | After Deallocation |
| --- | --- |
| 1 | 0 |
| 2 | 0 |
| 3 | 1872924688 |
| 4 | 21862 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 10 |

# Find the output of the following C++ Program

```
1  #include <iostream>
2  using namespace std;
3
4  void change(int* b);
5  int main()
6   { int a[5]={20,30,40,50,60};
7      change(a);
8      for (int i=4; i>=0; i--)
9       cout<<a[i]<<endl;
10      return 0;
11  }
12
13 void change(int* b)
14 { for (int i=0; i<4; i++)
15     {   *b=*b+1;
16       b++;
17     }
18 }
19
```

https://ideone.com/e0t05w

stdout

```
60
51
41
31
21
```

```
1   #include <iostream>
2   using namespace std;
3
4   int main()
5    { int a[5]={20,30,40,50,60};
6           for (int i=0; i<4; i++)
7       {    *a=*a+1;
8         a++;
9        }
10  for (int i=4; i>=0; i--)
11       cout<<a[i];
12       return 0;
13   }
14
```

https://ideone.com/BIqwFW

## compilation info

```
prog.cpp: In function ‘int main()’:
prog.cpp:8:5: error: lvalue required as increment operand
    a++;
     ^~
```

# Two Dimensional Arrays

- In C++

Static Array: `int  arr_2d[10][12];`

Dynamic Array:

```
  int **A;
A= new int*[rows];
 for (int i=0; i<rows; i++)
  A[i]=new int [cols];
```

- Two-dimensional arrays need not be rectangular. Each row can be a different length. Here's an example:

```
int **A;
A= new int*[rows];
A[0] = new int [1]; // A's first row has 1 column
A[1] = new int [2];  // A's second row has 2 columns
A[2] = new int [3];  // A's third row has 3 columns
A[3] = new int [5];  // A's fourth row has 5 columns
A[4] = new int [5];  // A's fifth row also has 5 columns
```

# Character Array (C++)

```
char str[10]="IITRoorkee";
    for (int i=0; i<10; i++)
    {     cout<<str+i;
          cout<<*(str+i);
  }
```

How to print the base address of static character array?

```
          (int *) str
          &str
```

https://ideone.com/sdAiT1

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7    char s[10]="IITR";
8    cout<<s<<endl;
9    cout<<&(*s)<<endl;
10   cout<<&s<<endl;
11   cout<<(int *) s;
12 }
13
```

https://ideone.com/uT8wj6

stdout

```
IITR
IITR
0x7ffc7794e8ae
0x7ffc7794e8ae
```

# Character String (C++)

```
Char* str="IITRoorkee";
    for (int i=0; i<10; i++)
    {      cout<<str+i;
           cout<<*(str+i);
  }
```

How to print the base address of dynamic character array?

```
           (int *) str
```

https://ideone.com/sTaqZ5

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7    char *s="IITR";
8    cout<<s<<endl;
9    cout<<&(*s)<<endl;
10   cout<<&s<<endl;
11   cout<<(int *) s;
12 }
13
```

https://ideone.com/nwFdbc

stdout

```
IITR
IITR
0x7ffd660ef360
0x55d14d7d5004
```

# What is the output of following strange code? Why?

```
int a[10];
cout<<a<<endl;
cout<<&a;
```

```
int a[10];
cout<<a+0<<endl;
cout<<&(a+0);
```

https://ideone.com/KWZeBw

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7     int a[1]={10};
8     cout<<a<<endl;
9     cout<<&a<<endl;
10 }
11
```

https://ideone.com/tARsQ5

stdout

```
0x7ffe27752e24
0x7ffe27752e24
```

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      // your code goes here
6  int a[10];
7
8      cout<<a<<endl;
9      cout<<&(a+0);
10 }
11
```

https://ideone.com/KWZeBw

## compilation info

```
prog.cpp: In function ‘int main()’:
prog.cpp:9:13: error: lvalue required as unary ‘&’ operand
   cout<<&(a+0);
             ^
```

# Structures in C++

https://ideone.com/Eu8HD2

```
1  #include <iostream>
2  using namespace std;
3
4  // Define a structure to represent time
5  struct Time {
6      int hours;
7      int minutes;
8      int seconds;
9  };
10
11 // Function to add two times
12 Time addTimes(Time t1, Time t2) {
13     Time sum;
14
15     sum.seconds = t1.seconds + t2.seconds;
16     sum.minutes = t1.minutes + t2.minutes;
17     sum.hours = t1.hours + t2.hours;
18
```

```
19          // Adjust for overflow
20          if (sum.seconds >= 60) {
21              sum.seconds -= 60;
22              sum.minutes++;
23          }
24          if (sum.minutes >= 60) {
25              sum.minutes -= 60;
26              sum.hours++;
27          }
28
29          return sum;
30      }
31
```

```
32  int main() {
33      Time time1, time2, result;
34
35      // Input for the first time
36      cout << "Enter the first time (hours minutes seconds): ";
37      cin >> time1.hours >> time1.minutes >> time1.seconds;
38
39      // Input for the second time
40      cout << "Enter the second time (hours minutes seconds): ";
41      cin >> time2.hours >> time2.minutes >> time2.seconds;
42
43      // Add the times
44      result = addTimes(time1, time2);
45
```

```
46      // Display the result
47      cout << "Sum of times: " << result.hours << " hours, "
48      << result.minutes << " minutes, " << result.seconds << " seconds" << endl;
49
50      return 0;
51  }
```

stdin

```
7 40 31
6 21 30
```

Enter the first time (hours minutes seconds): Enter the second time (hours minutes seconds): Sum of times: 14 hours, 2 minutes, 1 seconds

- **Doubts**

# Void pointer

```c
#include <stdio.h>

int main()
{
    int m=1000;
    void *ptr;
    ptr=&m;
    printf("Value which is pointed by x pointer : %d",*ptr);

     return 0;
}

```

compilation info

```
prog.c: In function ‘main’:
prog.c:8:54: warning: dereferencing ‘void *’ pointer
    printf("Value which is pointed by x pointer : %d",*ptr);
                                                     ^~~~
prog.c:8:54: error: invalid use of void expression
```

stdout

```
Standard output is empty
```

# Type casting in pointer

```c
#include <stdio.h>

int main()
{
    int m=1000;
    void *ptr;
    ptr=&m;
    printf("Value which is pointed by x pointer : %d",*(int*)ptr);

     return 0;
}

```

https://ideone.com/y2dfJt

stdout

```
Value which is pointed by x pointer : 1000
```

# Why we use void pointers?

https://ideone.com/EkpNGp

```c
#include<stdio.h>
int main()
{
  int a=1000; // initialization of a integer variable 'a'.
  float b=10.0; // initialization of a float variable 'b'.
  char c='B'; // initialization of a char variable 'c'.
   void *ptr; // declaration of void pointer.
   // assigning the address of variable 'a'.
   ptr=&a;
   printf("value of 'a' is : %d",*((int*)ptr));
   // assigning the address of variable 'b'.
   ptr=&b;
   printf("\nvalue of 'b' is : %f",*((float*)ptr));
   // assigning the address of variable 'c'.
   ptr=&c;
    printf("\nvalue of 'c' is : %c",*((char*)ptr));
    return 0;
}

```

## stdout

```
value of 'a' is : 1000
value of 'b' is : 10.000000
value of 'c' is : B
```

# calloc (Contiguous Allocation)

- calloc stands for "contiguous allocation."
- It allocates a block of memory for an array of elements, initializing all bits to zero.
- It ensures that the allocated memory is zero-initialized.
- It takes two arguments: the number of elements to allocate memory for and the size of each element in bytes.

# calloc

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

int main() {
    int n = 10;  // Number of integers to allocate
    int *arr;

    // Allocate memory for n integers
    arr = (int *)calloc(n, sizeof(int));

    // Check if memory allocation was successful
    if (arr == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }
```

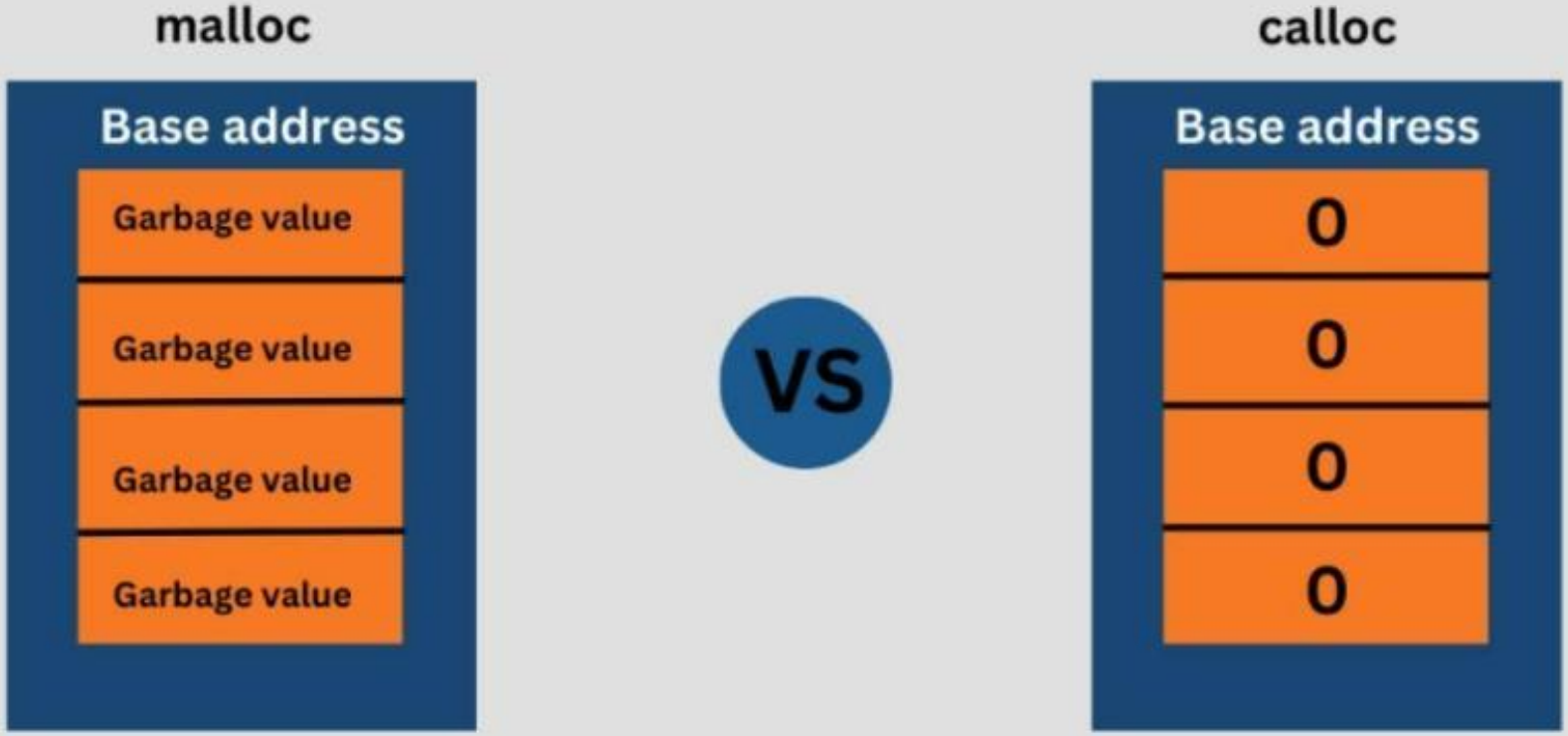
https://ideone.com/ATUIvD

```
16
17      // Use the allocated memory (for example, initialize the array)
18      for (int i = 0; i < n; i++) {
19          arr[i] = i * 10;
20          printf("%d ", arr[i]);
21      }
22
23      // Free the allocated memory (optional)
24      free(arr);
25
26      return 0;
27  }
28
```

stdout

```
0 10 20 30 40 50 60 70 80 90
```

# Difference Between malloc() and calloc() with Examples



- Difference between function pointers and function

# Function Pointers

```
1   #include<stdio.h>
2   int *larger(int *, int *);
3
4   int main() {
5       int a = 10, b = 15;
6       int *greater;
7       // passing address of variables to function
8       greater = larger(&a, &b);
9       printf("Larger value = %d", *greater);
10      return 0;
11  }
12
```

https://ideone.com/6TUUGe

# Function Pointers

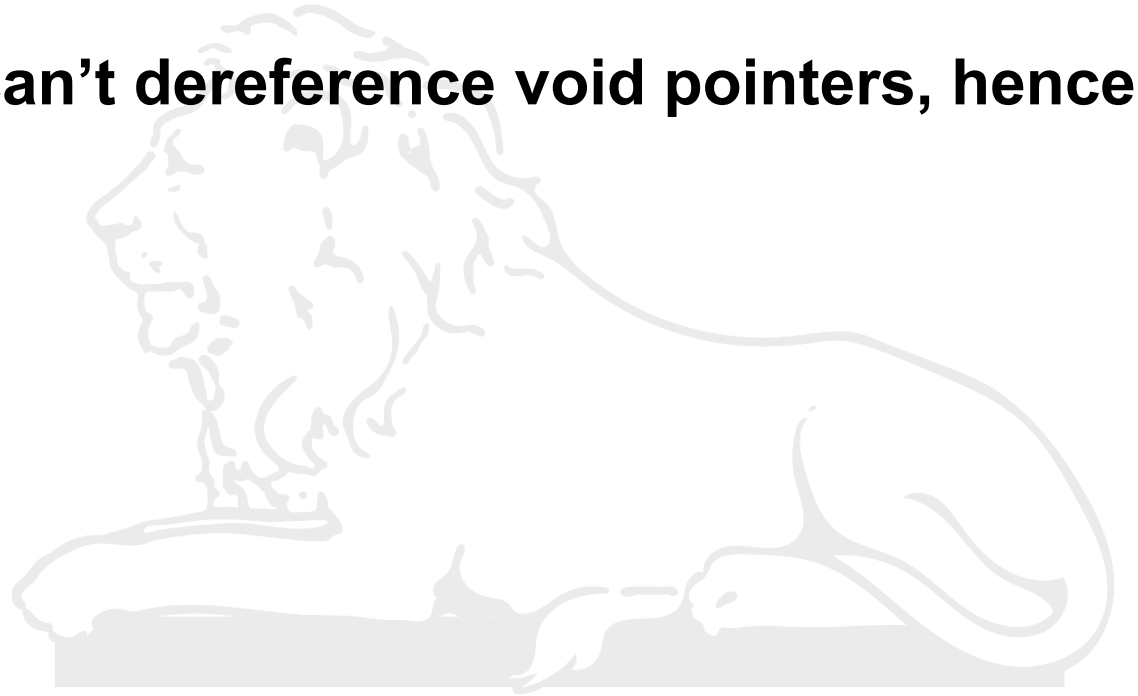
```
13  int *larger(int *a, int *b) {
14      if (*a > *b) {
15          return a;
16      }
17      // returning address of greater value
18      return b;
19  }
20
```

stdout

```
Larger value = 15
```

- Why Type casting is required for void pointers only while printing

**Ans: You can’t dereference void pointers, hence.**



# Scope of the variable

```
1   #include <iostream>
2   using namespace std;
3
4   int main() {
5       // your code goes here
6       {int a=10;
7       cout<<a;
8       }
9       cout<<a;
10      return 0;
11  }
12
```

# Static variable

- An ordinary variable is limited to the scope in which it is defined, while the scope of the static variable is throughout the program.

# Static variable

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

int func()
{
    int count=0; // variable initialization
    count++; // incrementing counter variable
    return count;
}
```

stdout

```
1
1
```

https://ideone.com/kibY8C

# Static variable

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

int func()
{
    static int count=0; // static
    count++; // incrementing counter variable
    return count;
}
```

stdout

```
1
2
```

https://ideone.com/sgUF2z

# Addition of 2 Distances

https://ideone.com/3Bz7fU

```c
#include <stdio.h>
// Structure to represent distance
typedef struct {
    int feet;
    int inches;
} Distance;
// Function to add two distances
Distance addDistances(Distance d1, Distance d2) {
    Distance result;
    result.feet = d1.feet + d2.feet;
    result.inches = d1.inches + d2.inches;

    // Adjust inches to be less than 12
    if (result.inches >= 12) {
        result.inches -= 12;
        result.feet++;
    }

    return result;
}

```

```c
int main() {
    Distance distance1, distance2, sum;

    // Input first distance
    printf("Enter first distance (feet inches): ");
    scanf("%d %d", &distance1.feet, &distance1.inches);

    // Input second distance
    printf("Enter second distance (feet inches): ");
    scanf("%d %d", &distance2.feet, &distance2.inches);

    // Add the distances
    sum = addDistances(distance1, distance2);

    // Display the sum
    printf("Sum of distances: %d feet %d inches\n", sum.feet, sum.inches);

    return 0;
}
```

stdin

```
7 8
6 7
```

stdout

```
Enter first distance (feet inches): 7 8
Enter second distance (feet inches): 6 7
Sum of distances: 14 feet 3 inches
```

- Do we have any specific application of pointers that cannot be done otherwise ?

Dynamic memory allocation

# Thank You!