# CS F111: Computer Programming

**(Second Semester 2020-21)**

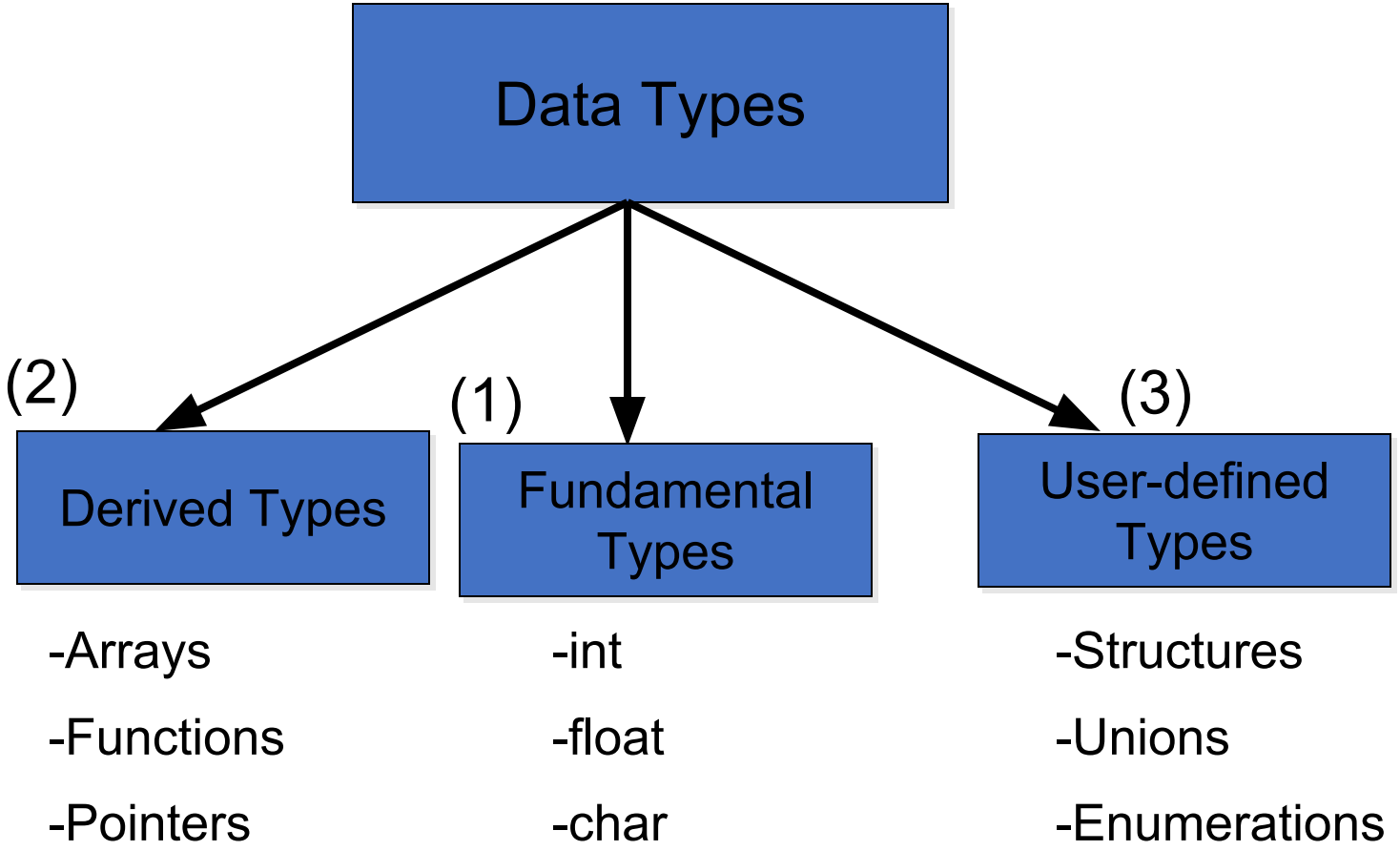
## Lect 14: Arrays:Intro

Dr. Nikumani Choudhury
Asst. Prof., Dept. of Computer Sc. & Information Systems
nikumani@hyderabad.bits-pilani.ac.in

**BITS** Pilani

Hyderabad Campus

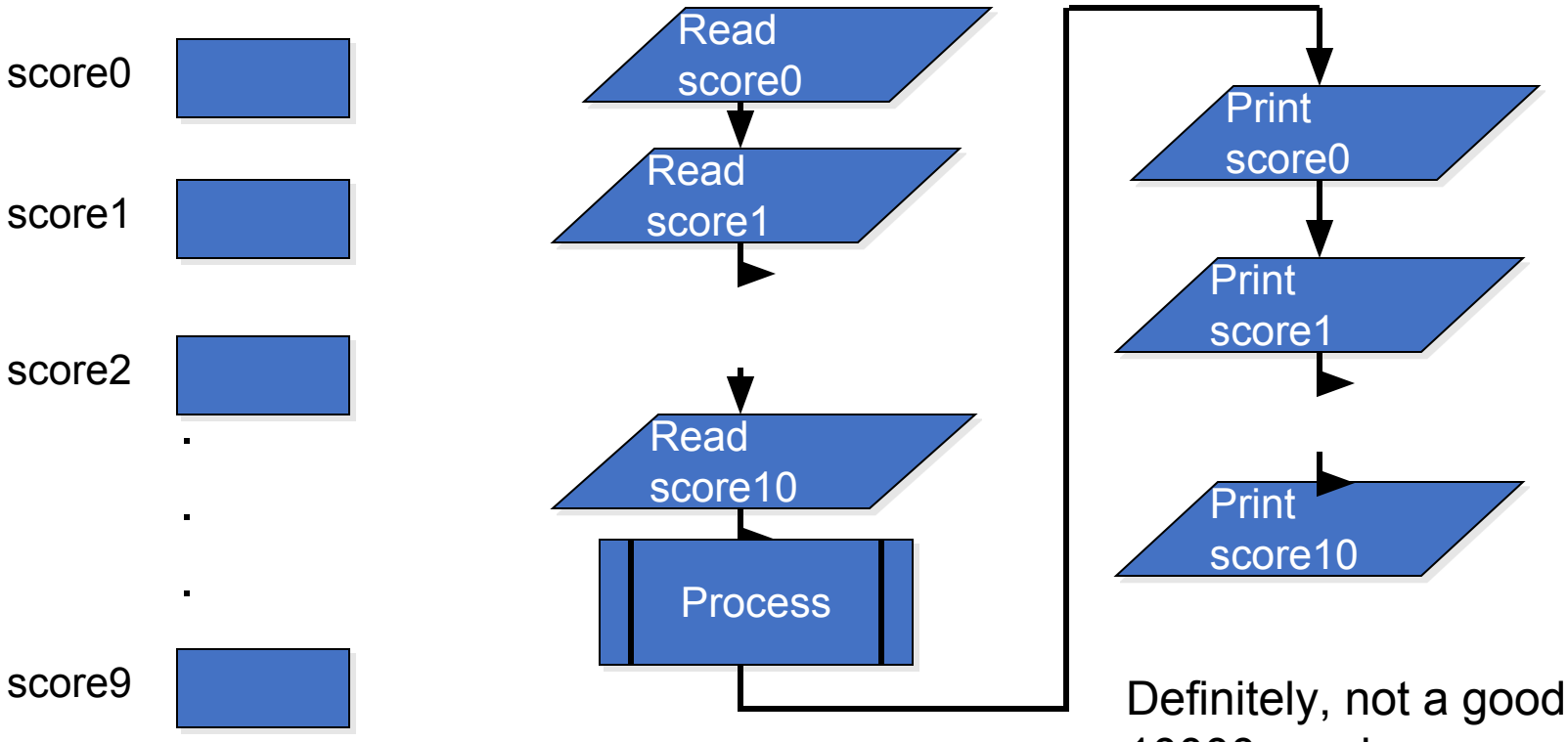
# Arrays




Data Structure: Arrays, Structures, Lists, Stacks, Trees etc.

# Concepts

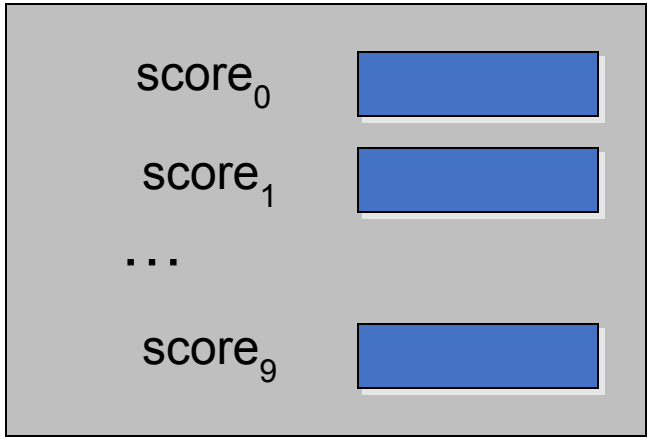
- Let us assume that we have to read, process and print 10 numbers (integers) and keep those in the memory throughout the execution.



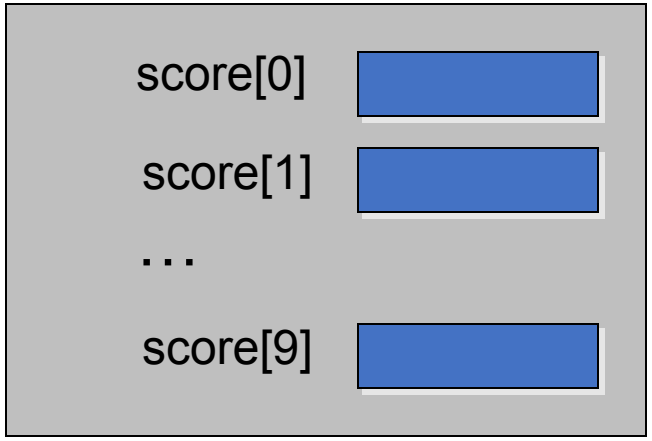

Definitely, not a good idea for 10000 numbers

# Concepts continued…

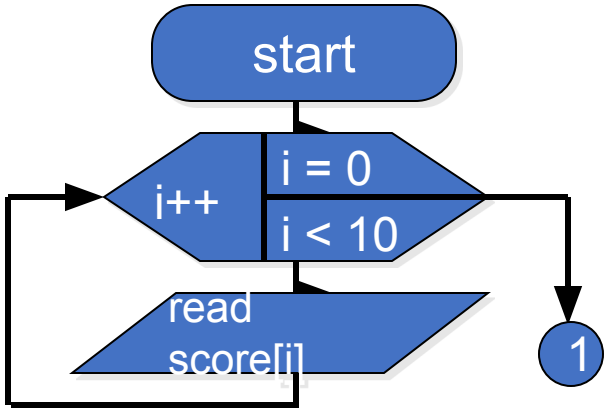
- An array of scores: Only one variable is enough

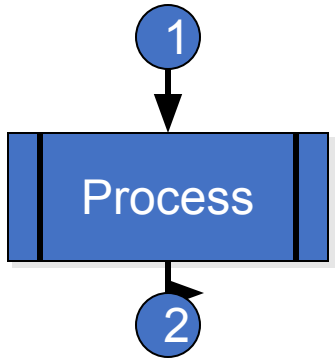


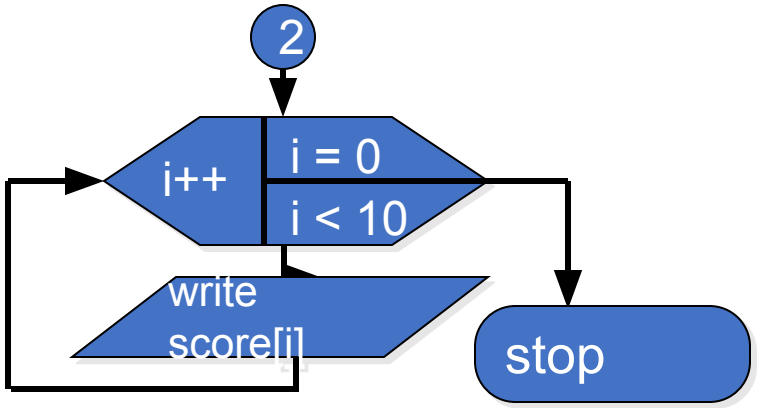

Subscripting




Indexing



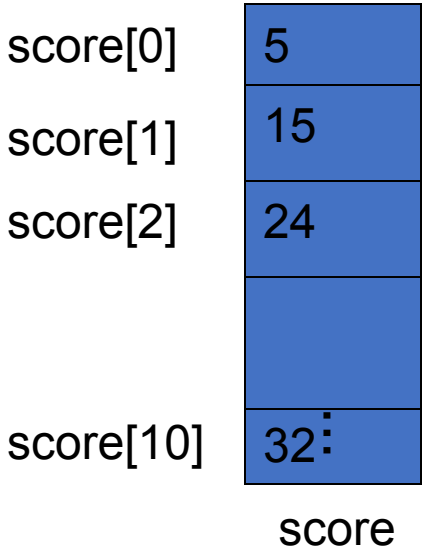



# Arrays in C

- An array is a sequenced collection of elements of the same data type addressable by index.

- It is a convenient data structure for representing large number of <u>homogenous</u> values.

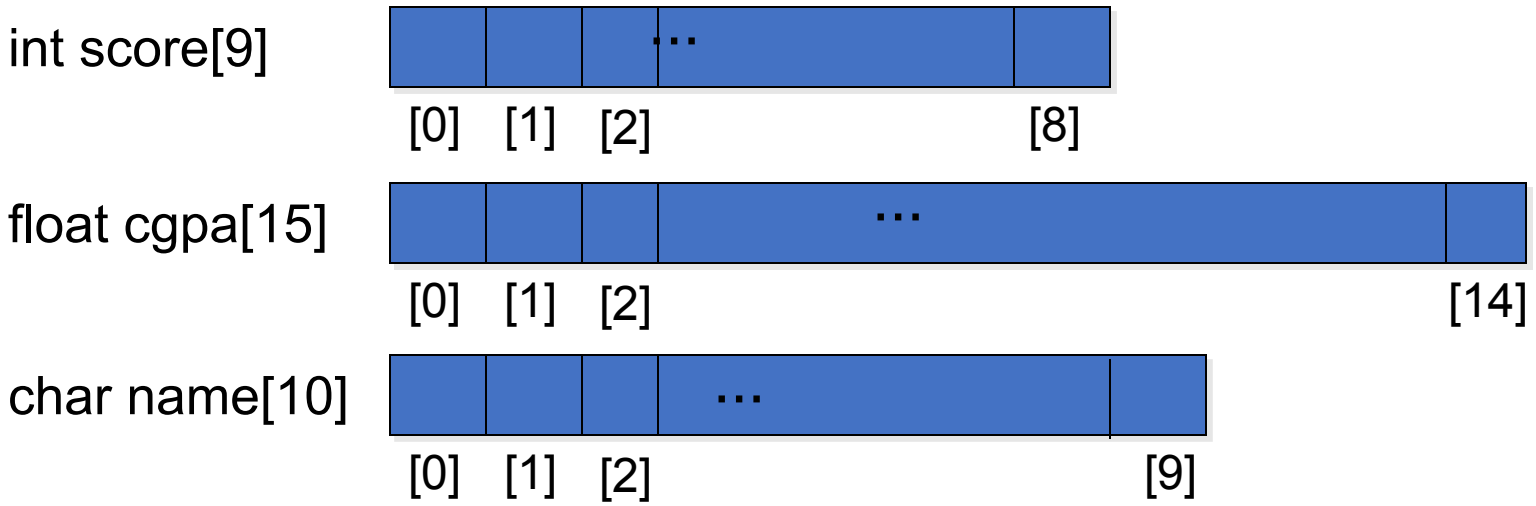
- Examples:
    - List of marks of students
    - List of grades of students
    - List of names of students



# Array declaration and definition

- Types: Fixed length and variable length
  - when the program is written and when it is run
- Declaration format:
  - type arrayName [arraySize]
- Three different <u>fixed length</u> arrays:

```
int a[5];
a = { 1, 2, 3, 4, 5 };
```




- Array size is variable in variable length arrays:( runtime)

# Storing values in Arrays continued…

- more examples on <u>Initialization</u>:

```
#include <stdio.h>
main()
{
  int i;
  int max[5] = {8, 7};  /*auto*/
  for (i=0;i<=7;i++)
    printf ("%d\n", max[i]);
  return 0;
}
```

```
bash$ ./a.out
8
7
0
0
0
5
-1076194544
-1076194472
bash$
```

```
static int max[5] = {8,7};
```

```
bash$ ./a.out
8
7
0
0
0
0
0
0
bash$
```

# Storing values in Arrays continued…

- <u>Inputting values:</u>

```
for ( i = 0; i < 9; i++)
    scanf ("%d", &score [i]);
```

For filling completely

- If all the elements are not to be filled, use event controlled loops like while or do…while

<u>Assigning values:</u>

```
score [3] = 34;
```

```
for (i = 0; i < 25; i++)
    second [i] = first [i];
```

Copying elements

```
for (i = 0; i < 9; i++)
    score [i] = i*2;
```

Values follow a pattern

# Storing values in Arrays continued…

Exchanging values:

number[ ];

| 3 | 7 | 12 | 24 | 45 |
|---|---|---|---|---|
| [0] | [1] | [2] | [3] | [4] |

number[3] = number [1];

| 3 | 7 | 12 | 7 | 45 |
|---|---|---|---|---|
| [0] | [1] | [2] | [3] | [4] |

number[1] = number [3];

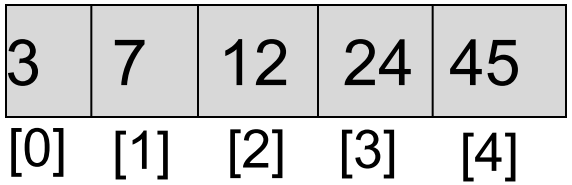
| 3 | 7 | 12 | 7 | 45 |
|---|---|---|---|---|
| [0] | [1] | [2] | [3] | [4] |

wrong way

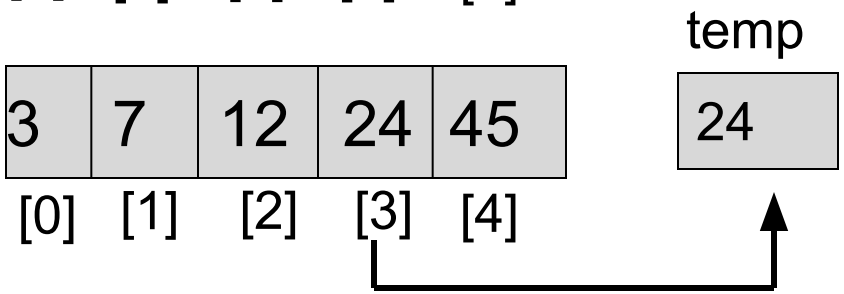
# Storing values in Arrays continued…

Exchanging values continued:

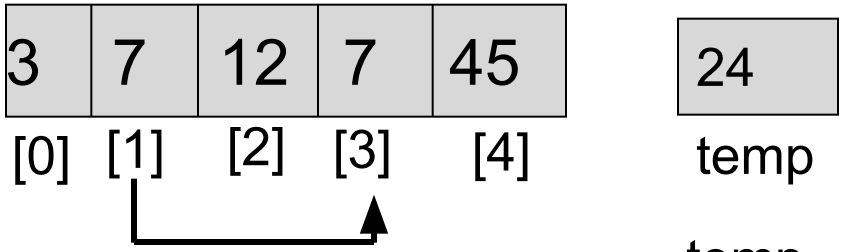
number[ ];



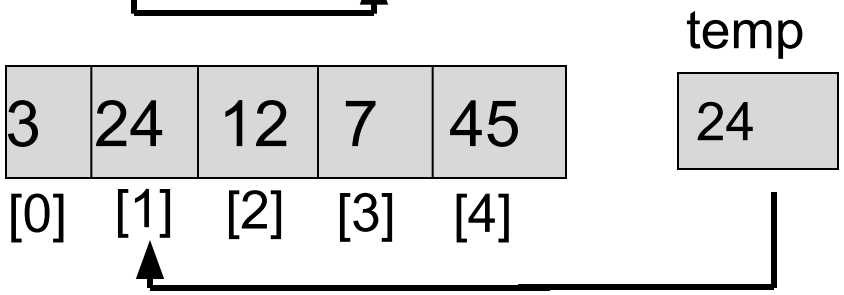

temp = number[3];




number[3] = number [1];




number[1] = temp;




Correct way

# Example 1

- Let us run a C program to evaluate:

$$\text{Sum} = \sum_{k=0}^{10} x_k$$

- using arrays at onlinegdb.

- **Modify** to compute the below Average:

$$\text{Avg} = \sum_{i=1}^{n} x_i^2 \Big/ n$$

# Example 2:

Take 10 numbers, find out how many are >= average of those numbers.

```c
#include <stdio.h>
#define MAX_SIZE 25

int main()
{

 int i, sum, avg, count=0, a[10];

 printf ("Enter the numbers");
 for (i= 0; i <=9; i++)
 {
     scanf("%d", & a[i]);
     sum = sum + a[i];

 }
 avg = sum/10;

 for (i=0; i <= 9; i++)
 {
     if (a[i] >= avg )
     {
         count++;
     }
 }
 printf ("%d", count);
 return 0;
}
```

# Printing values: Example 3

```
for (i = 0; i < 9; i++)
   printf ("%d", score [i]);
printf ("\n");
```

Results:

```
 1  2  3  4  5  6  7  8  9  10
21 22 23 24 25 26 27 28 29 30
41 42 43 44 45
```

```
#define MAX_SIZE 25
main( )
{
int list [MAX_SIZE] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                21, 22, 23, 24, 25, 26, 27, 28,                29,
30, 41, 42, 43, 44, 45};
        int numPrinted;
        numPrinted = 0;
        for (int i = 0; i < MAX_SIZE; i++)
           {
            printf("%3d", list[i]);
            if (numPrinted < 9)
                numPrinted++;
            else
               {
                printf("\n");
                numPrinted = 0;
                }
           }
        return 0;
}       // main
```

# Precedence of Array references

number[ ];

| 3 | 7 | 12 | 24 | 45 |
|---|---|---|---|---|
| [0] | [1] | [2] | [3] | [4] |

number [3] = number [4] + 15;

number [3] = 45 + 15;

# Index Range Checking

- C does not check the boundary of an array.
- It is programmers responsibility to check that all references are valid and within the range.
- Using an invalid index could cause unpredictable results.

```
for (i = 1; i <= 9; i++)
    scanf ("%d", &score [i]);
```

Erroneously started at 1 in place of 0

```
main()
 {
  int i, num [40];
  for (i = 0; i <= 100; i++)
       num [ i ] = i;
 }
```

Suicidal: may hang or give unpredictable results.

# Example 4

```
/*Read a number series and print it
reversed */
#include <stdio.h>

int main ( )
{
  int readNum;
  int numbers[50];
  printf(“Enter up to 50 integers:\n");
  printf("How many would you? ");
  scanf ("%d", &readNum);

  if (readNum > 50)
      readNum = 50;
  printf("Enter your numbers: \n");

  for (int i = 0; i < readNum; i++)
     scanf("%d", &numbers[i]);

  printf("Numbers reversed are: \n");
  for (int i=readNum-1,numPrinted=0; i>= 0; i--)
    {
      printf("%3d", numbers[i]);
      if (numPrinted < 9)
      numPrinted++;
      else
        {
      printf("\n");
      numPrinted = 0;
        }
    }
  return 0;
}
```

Results:
Enter up to 50 integers:
How many would you? 12
Enter your numbers:
1 2 3 4 5 6 7 8 9 10 11 12
Numbers reversed are:
12 11 10 9 8 7 6 5 4 3
2 1