CS1100 – Introduction to Programming

Trimester 3, April – June 2021

Instructor:

Shweta Agrawal (shweta.a@cse.iitm.ac.in)

Lecture 15

CS1100 – Introduction to Programming

- Programming : From Turtle to C.
- Data Types in C, Operators. Input and the Output.
- Modifying the control flow in Programs `if-else`, `switch`, loops : `while`, `do-while`, `for`.

} So far ...

CS1100 – Introduction to Programming

- Programming : From Turtle to C.
- Data Types in C, Operators. Input and the Output.
- Modifying the control flow in Programs `if-else`, `switch`, loops : `while`, `do-while`, `for`.

} So far ...

- `One-dimensional Arrays` in C.

} Coming Up

Programming for Real life problems

Here are some real life problems that we may want to solve using computers.

- Given the marks obtained by students in a class, print out the marks in the non-decreasing order. That is, smallest marks first.

Programming for Real life problems

Here are some real life problems that we may want to solve using computers.

- Given the marks obtained by students in a class, print out the marks in the non-decreasing order. That is, smallest marks first.
- Given a road map of India find the shortest path from Main gate of IIT Madras to Dharwad.

Programming for Real life problems

Here are some real life problems that we may want to solve using computers.

- Given the marks obtained by students in a class, print out the marks in the non-decreasing order. That is, smallest marks first.
- Given a road map of India find the shortest path from Main gate of IIT Madras to Dharwad.
- Given the positions, velocities and masses of stars, determine their state 1 million years from today.

Programming for Real life problems

Here are some real life problems that we may want to solve using computers.

- Given the marks obtained by students in a class, print out the marks in the non-decreasing order. That is, smallest marks first.
- Given a road map of India find the shortest path from Main gate of IIT Madras to Dharwad.
- Given the positions, velocities and masses of stars, determine their state 1 million years from today.

Difficulties : Size of the input data is huge !

Programming for Real life problems

Here are some real life problems that we may want to solve using computers.

- Given the marks obtained by students in a class, print out the marks in the non-decreasing order. That is, smallest marks first.
- Given a road map of India find the shortest path from Main gate of IIT Madras to Dharwad.
- Given the positions, velocities and masses of stars, determine their state 1 million years from today.

Difficulties : Size of the input data is huge !
See example 1 : defining a variable for each mark is not feasible

What is an array?

- To store several elements of the same type.
  - store 100 integers.
  - store 2000 characters.
  - store 500 floats.

What is an array?

- To store several elements of the same type.
  - store 100 integers.
  - store 2000 characters.
  - store 500 floats.

- **Declaration :**

```
data-type array-name[array-size];
```

What is an array?

- To store several elements of the same type.
  - store 100 integers.
  - store 2000 characters.
  - store 500 floats.

- **Declaration :**

```
data-type array-name[array-size];
```

- `int marks[7];`

What is an array?

- To store several elements of the same type.
  - store 100 integers.
  - store 2000 characters.
  - store 500 floats.

- **Declaration :**
  `data-type array-name[array-size];`
- `int marks[7];`
- `char name[10];`
- `float score[1000];`

What is an array?

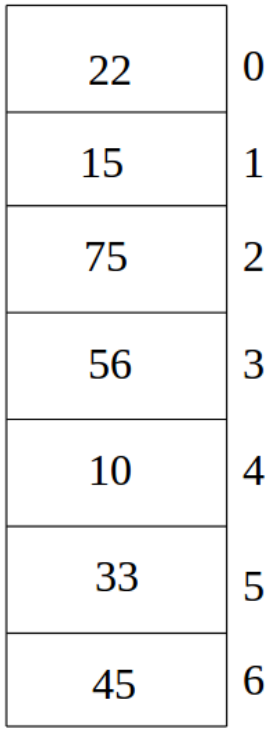
- To store several elements of the same type.
  - store 100 integers.
  - store 2000 characters.
  - store 500 floats.

- **Declaration :**
  `data-type array-name[array-size];`
- `int marks[7];`
- `char name[10];`
- `float score[1000];` - defines 1000 variables!

What is an array?

- To store several elements of the same type.
  - store 100 integers.
  - store 2000 characters.
  - store 500 floats.

- **Declaration :**
  `data-type array-name[array-size];`
- `int marks[7];`
- `char name[10];`
- `float score[1000];` - defines 1000 variables!
- the value of `marks[2]` is 75.
- new values can be assigned to elements
  `marks[3] = 36;`



Storing Arrays

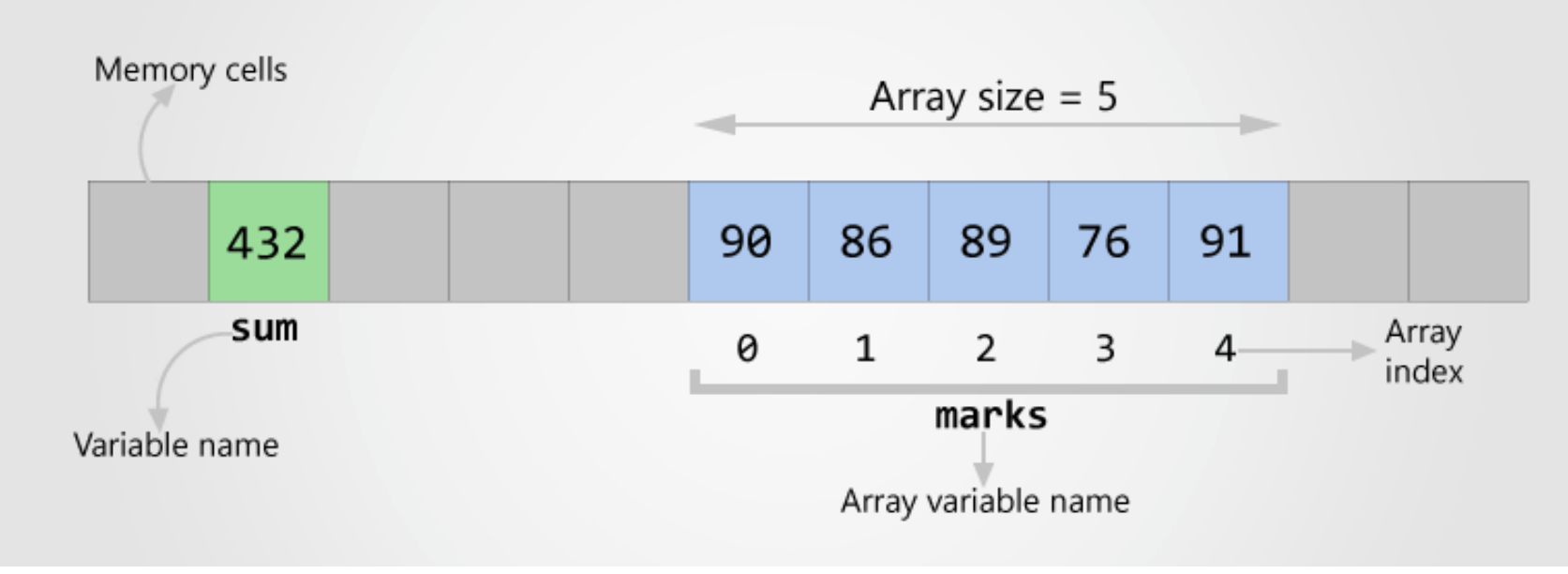
- All elements are of same type.

Storing Arrays

- All elements are of same type.
- The number of elements is finite and fixed !.

Storing Arrays

- All elements are of same type.
- The number of elements is finite and fixed !.
- Elements are stored in contiguous memory locations.



Simple use of arrays

Grading for an exam is over and we wish to plot a histogram of marks of the students.

Simple use of arrays

Grading for an exam is over and we wish to plot a histogram of marks of the students.

- Assume test was for 25 marks.
- Integer scores. No negative marking.
- What are the possible different scores?

Simple use of arrays

Grading for an exam is over and we wish to plot a histogram of marks of the students.

- Assume test was for 25 marks.
- Integer scores. No negative marking.
- What are the possible different scores?
- Use arrays instead of 25 different variables.

Simple use of arrays

Grading for an exam is over and we wish to plot a histogram of marks of the students.

- Assume test was for 25 marks.
- No negative marking.
- What are the possible different scores?
- Use arrays instead of ~~25~~ 26 different variables.

Counting number of students who scored marks-i

```
#include<stdio.h>
main() {
     const int MAX_MARKS = 25;
     const int NUM_STUDENTS = 56;
     int marksCount[MAX_MARKS+1];

     int i, currMarks;
     for (i=1; i<= NUM_STUDENTS; i++) {
         printf("Enter the marks for Rollnumber %d\t", i);
         scanf("%d", &currMarks);
         marksCount[currMarks]++;
     }
}
```

Counting number of students who scored marks-i

```
#include<stdio.h>
main() {
      const int MAX_MARKS = 25;
      const int NUM_STUDENTS = 56;
      int marksCount[MAX_MARKS+1];

      int i, currMarks;
      for (i=1; i<= NUM_STUDENTS; i++) {
          printf("Enter the marks for Rollnumber %d\t", i);
          scanf("%d", &currMarks);
          marksCount[currMarks]++;
      }
}
```

Is the program correct?

Counting number of students who scored marks-i

```
#include<stdio.h>
main() {
      const int MAX_MARKS = 25;
      const int NUM_STUDENTS = 56;
      int marksCount[MAX_MARKS+1];

      int i, currMarks;
      for (i=1; i<= NUM_STUDENTS; i++) {
          printf("Enter the marks for Rollnumber %d\t", i);
          scanf("%d", &currMarks);
          marksCount[currMarks]++;
      }
}
```

Is the program correct?

- Initialization of `marksCount` missing.
- What if the user enters marks outside the range?

Counting number of students who scored marks-i

```
#include<stdio.h>
int main() {
     const int MAX_MARKS = 25;
     const int NUM_STUDENTS = 5;
     int marksCount[MAX_MARKS+1];

     int i, currMarks;
     int sum;

     for (i=0; i<= MAX_MARKS; i++) {
         marksCount[i] = 0;
     }

     for (i=1; i<= NUM_STUDENTS; i++) {
         printf("Enter the marks for Rollnumber %d\t", i);
         scanf("%d", &currMarks);
         if (currMarks >= 0 && currMarks <= MAX_MARKS) {
            marksCount[currMarks]++;
         }
     }
```

Initializing an array

Different ways of initializing array.

- int count[] = {10, 23, 50};
  Creates an array of 3 integers. count[0], count[1], count[2].
- int count[10] = {0};

Initializing an array

Different ways of initializing array.

- int count[] = {10, 23, 50};
  Creates an array of 3 integers. count[0], count[1], count[2].
- int count[10] = {0};
- Using a loop to explicitly initialize the elements.

Initializing an array

Different ways of initializing array.

- int count[] = {10, 23, 50};
  Creates an array of 3 integers. count[0], count[1], count[2].
- int count[10] = {0};
- Using a loop to explicitly initialize the elements.

Common Mistake: Forgetting to initialize the elements of array.

Evaluating a polynomial

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- n is the degree of a polynomial.
- User provides n coefficients.
- User provides the value of x at which polynomial has to be evaluated.
- Evaluate the polynomial.

Evaluating a polynomial

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- Evaluate each term separately.
  - n additions.
  - $n + (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n+1)}{2}$ multiplications.

Evaluating a polynomial

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- Evaluate each term separately.
  - n additions.
  - $n + (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n+1)}{2}$ multiplications.

- $P(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-2} + x(a_{n-1} + xa_n))\ldots)$
  - n additions.
  - n multiplications.

Evaluating a polynomial

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

int main() {
    int x, n, i;
    int coeff[20]; // maximum degree = 20.
    int value = 0;
    int product = 1;

    scanf("%d %d", &n, &x);

    for (i=0; i<=n; i++) {
        scanf("%d", &coeff[i]);
        product = coeff[i]* pow(x, i);
        value = value + product;
    }

    printf("%d\n", value);
    return 0;
}
```

Evaluating a polynomial

```
#include<stdio.h>

main() {
    int x, n, i;
    int coeff[20]; // maximum degree = 20.
    int value;

    scanf("%d %d", &n, &x);

    for (i=0; i<=n; i++) {
        scanf("%d", &coeff[i]);
    }


    /* Fill in your code here */


    printf("%d\n", value);
}
```

Evaluating a polynomial

Character arrays

char name[20];

Different ways of initialization

- char name[20] = “Avani”;
- char name[20] = {‘A’, ‘V’, ‘A’, ‘N’, ‘I’, ’null char’};
- char name[20];
  scanf(“%s”, name);

Character arrays

char name[20];

Different ways of initialization

- char name[20] = “Avani”;
- char name[20] = {‘A’, ‘V’, ‘A’, ‘N’, ‘I’, ’null char’};
- char name[20];
  scanf(“%s”, name);
- char name[20];
  name = ”AVANI”; Incorrect!!

What is the output of this program?

```
#include<stdio.h>
int main() {
    char name[20] = "AVANI";
    int i;

    for (i=10; i<20; i++) {
        name[i] = 'X';
    }

    printf("name = %s\n", name);

    for (i=0; i<20; i++) {
        printf("%c %d\n", name[i], name[i]);
    }
    return 0;
}
```

Character arrays and standard library support

- Character arrays or strings occur very often.
- C provides a standard library string.h
- exposes several useful functions:
  - strlen
  - strcmp
  - strcpy
  - strstr

Compare two strings

User input two strings $s1, s2$. Determine if $s1$ and $s2$ are the same.

Compare two strings

User input two strings $s1, s2$. Determine if $s1$ and $s2$ are the same.

- if (s1 == s2) This does not work

Compare two strings

User input two strings $s1, s2$. Determine if $s1$ and $s2$ are the same.

- if (s1 == s2) This does not work
- Write your own string compare.
- Assume strlen is available from string.h

Palindromes

A string is a palindrome iff string == reverse(string)

Palindromes

A string is a palindrome iff string == reverse(string)

- malayalam
- neveroddoreven
- dontnod

Palindromes

A string is a palindrome iff string == reverse(string)

- malayalam
- neveroddoreven
- dontnod

Write a program to determine if the given string is a palindrome.