

Finding Roots : Find x such that $f(x) = 0$

Newton-Raphson Method :

Idea : Start with an approximation for the root. And then successively improve the value of the approximation.

From Taylor Series : $f(x) \approx f(x_0) + f'(x_0) \cdot (x - x_0)$

Choose x to be the root and x_0 to be the initial guess.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Finding Roots : Find x such that $f(x) = 0$

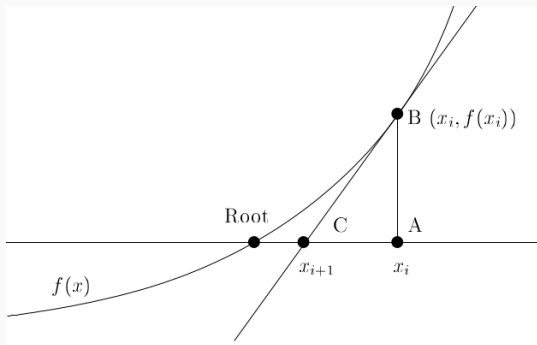
Newton-Raphson Method :

Idea : Start with an approximation for the root. And then successively improve the value of the approximation.

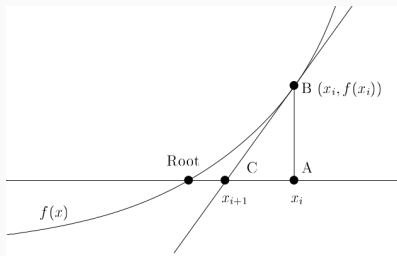
From Taylor Series : $f(x) \approx f(x_0) + f'(x_0) \cdot (x - x_0)$

Choose x to be the root and x_0 to be the initial guess.

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



Newton-Raphson Method : Reading the Figure



- $A = (x_i, 0)$ and $B = (x_i, f(x_i))$.
- Draw the tangent to the curve at B .
- The tangent meets the x -axis at $C = (x_{i+1}, 0)$.
- Use x_{i+1} as the next approximation to the root.

The tangent passes through B and C ,
so

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}.$$

Therefore,

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}.$$

Newton-Raphson Method : Program Segment

For

$$f(x) = x^2 - k,$$

$$f'(x) = 2x,$$

Newton's update
becomes

$$x_{i+1} = \frac{x_i + k/x_i}{2}.$$

```
// f(x) = x*x - k and f'(x) = 2x
#include "stdio.h"
#include "math.h"
int main()
{
    float k;
    scanf("%f",&k);
    float xi=1;
    while (fabs(xi*xi - k) > 0.001) {
        xi = (xi + k/xi)/2;
        printf("%f\n",xi);
    }
    return 0;
}
```

Roundup : From Loops to Roots

- **New programming constructs:**
 - `while`, `for`, `do-while` constructs in C.
 - `break`, `continue` statements in the body of the loop.
 - Several examples.
- **New ideas about program development:**
 - Stages of program development.
 - Correctness and termination of programs.
- **Programming - Servant of Science & Engineering:**
 - Programming to number theorists rescue - Eg: Collatz sequences, primality testing, gcd computation.
 - Programming to engineer's rescue - computing approximations to Taylor series expansions (eg: $\sin(x)$), finding approx roots of equations (eg: approximating square root).

Roundup : From Loops to Roots

- **New programming constructs:**
 - while, for, do-while constructs in C.
 - break, continue statements in the body of the loop.
 - Several examples.
- **New ideas about program development:**
 - Stages of program development.
 - Correctness and termination of programs.
- **Programming - Servant of Science & Engineering:**
 - Programming to number theorists rescue - Eg: Collatz sequences, primality testing, gcd computation.
 - Programming to engineer's rescue - computing approximations to Taylor series expansions (eg: $\sin(x)$), finding approx roots of equations (eg: approximating square root).

Coming up : More intense programming - functions, recursion, pointers, arrays.

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.

} Week 1-5

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.

} Week 1-5

-
- One-dimensional Arrays in C.

} Week 6

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;`

22	0
15	1
75	2
56	3
10	4
33	5
45	6

Storing Arrays

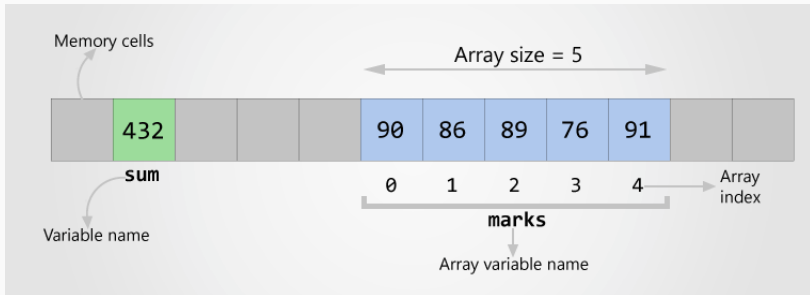
- All elements are of **same** type.

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.

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>

int main() {
    int numStudents, marks, i;
    int marksCount[26];

    scanf("%d", &numStudents);
    for (i = 0; i < numStudents; i++) {
        scanf("%d", &marks);
        marksCount[marks]++;
    }
    return 0;
}
```

Counting number of students who scored marks-i

```
#include <stdio.h>

int main() {
    int numStudents, marks, i;
    int marksCount[26];

    scanf("%d", &numStudents);
    for (i = 0; i < numStudents; i++) {
        scanf("%d", &marks);
        marksCount[marks]++;
    }
    return 0;
}
```

Is the program correct?