



Programming Fundamentals

LECTURE 01: REVISION
BY: ZUPASH AWAIS

WEEK 01

TOPICS FOR REVISION

Introduction to Computer System

Problem-Solving

Algorithm, Pseudocode, and Flow Chart

Variables

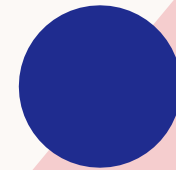
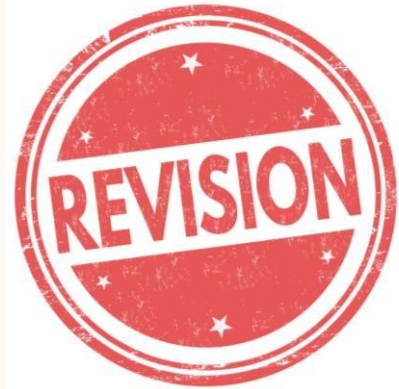
Data Types

Operators

Decision Control

Iterative Control

1D Arrays

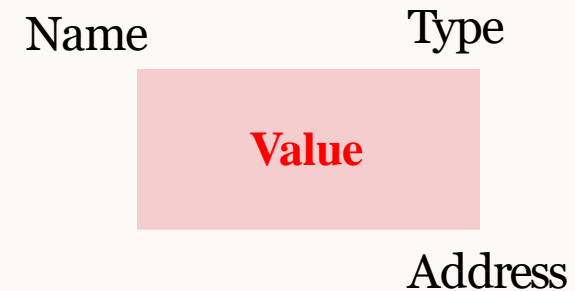


VARIABLES

To indicate the storage area, each variable should be given a unique name (identifier). For example,

```
age= 20;
```

Here age is the variable and 20 is the value stored in age variable



DATA TYPES

In C++, data types are declarations for variables. This determines the type and size of data associated with variables. For example,

```
int age = 20;
```

Here int is the datatype for variable age. Each datatype have its defined size. For example, int is of 4Bytes.

Built In Data Types	User Defined Datatypes
Int	Structure
Float	Class
Double	String (Collection of Characters)
Char etc	

TYPECASTING

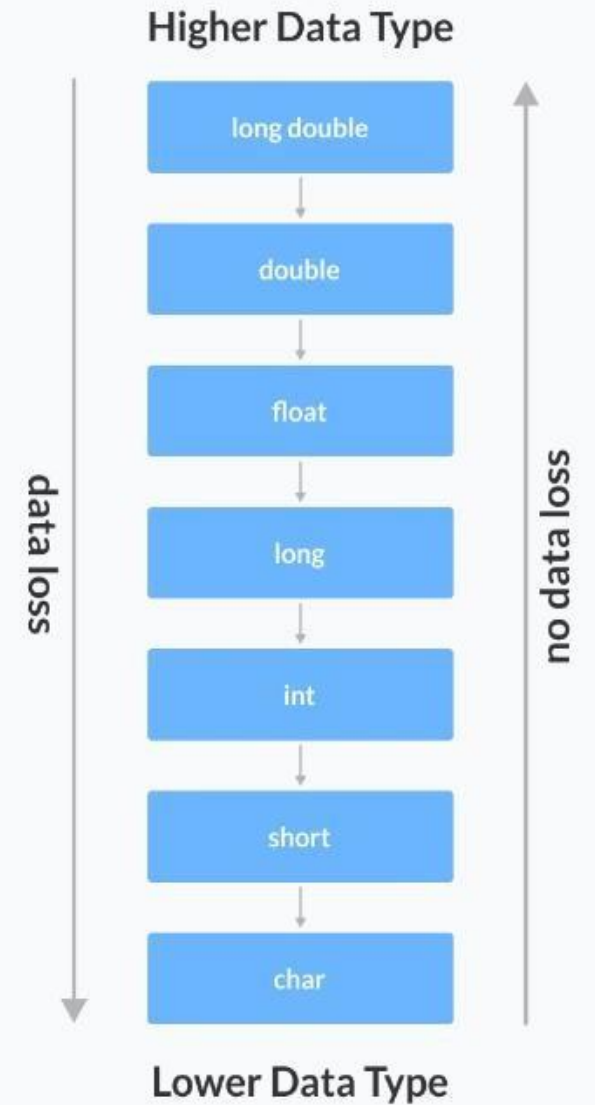
- Implicit
- Explicit

//Implicit Typecasting

```
#include<iostream>
using namespace std;
int main()
{
    int ch = 10;
    float a = 3, b;
    b = ch / a;
    cout << b << endl;
}
```

//Explicit Typecasting

```
#include<iostream>
using namespace std;
int main()
{
    int ch = 10, a = 3;
    float b;
    b = (float) ch / a;
    cout << b << endl;
}
```



OPERATORS

An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators and provide the following types of operators:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Misc. Operators

PREFIX AND POSTFIX

Prefix: is a notation that writes the operator before operands

Postfix: is a notation that writes the operator after the operands

- ++a
- a++
- --a
- a--

Example

```
int main()
{
    int i=10;
    cout<< (i++) + (++i);
}
```

DECISION CONTROLS

The decision control statements are the decision-making statements that decides the order of execution of statements based on the conditions. In the decision-making statements the programmer specify which conditions are to be executed or tested with the statements to be executed if the condition is true or false.

1. If statement
2. If/else statement
3. If/elseif/else statement
4. Switch statement
5. Nested if/else statement

CONDITION

```
int a = 10;
```

```
int b = 20;
```

```
cout<< (a>b); //the condition can be true or false, will return 1/0
```

```
cout<< (a==b);
```

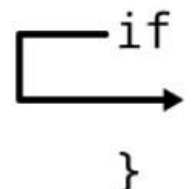
```
cout<< (a || b);
```

```
cout<< (a<b);
```

IF/ELSE

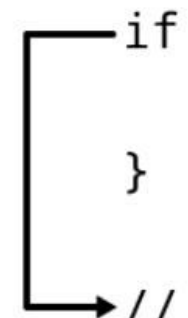
Condition is true

```
int number = 5;  
  
if (number > 0) {  
    // code  
}  
  
// code after if
```



Condition is false

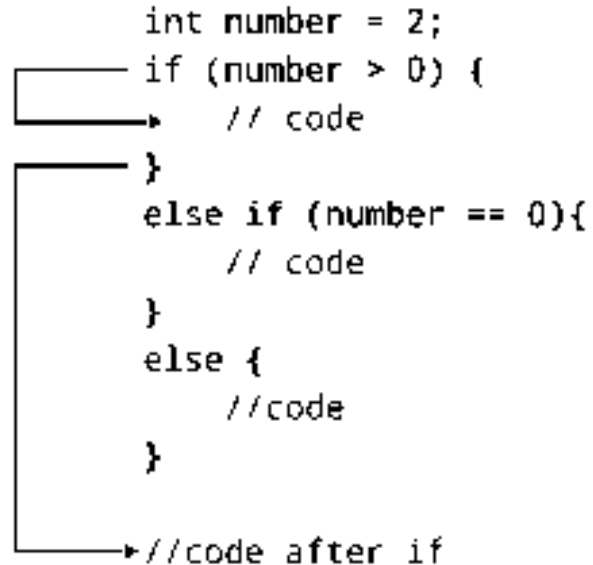
```
int number = 5;  
  
if (number < 0) {  
    // code  
}  
  
// code after if
```



IF/ELSE IF

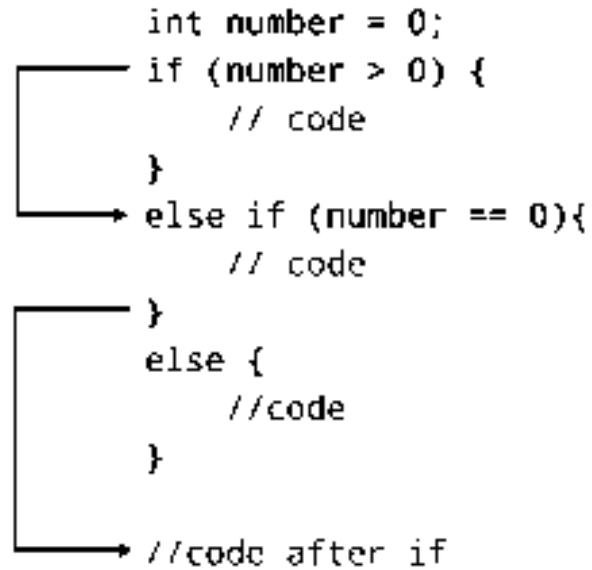
1st Condition is true

```
int number = 2;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
//code after if
```



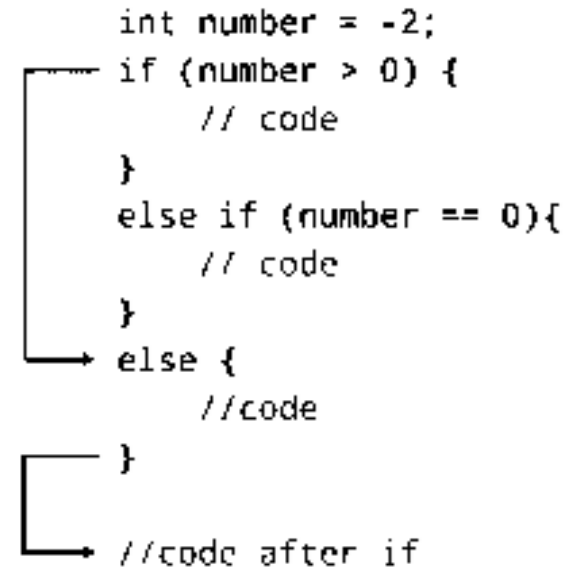
2nd Condition is true

```
int number = 0;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
//code after if
```



All Conditions are false

```
int number = -2;  
if (number > 0) {  
    // code  
}  
else if (number == 0){  
    // code  
}  
else {  
    //code  
}  
//code after if
```



SWITCH CASE

```
switch (expression)
{
    case constant1:
        // statements
        break;

    case constant2:
        // statements
        break;
    .
    .
    .
    default:
        // default statements
}
```

```
int a=10;
int b=20;

switch(a<b)
{
    case 1:
        cout<<"true";
        break;
    case 0:
        cout<<"false";
        break;
}
```

LOOP

- For
- While
- Do-while

```
#include <iostream>

using namespace std;

int main() {
    for (int i = 1; i <= 5; ++i) {
        cout << i << " ";
    }
    return 0;
}
```

```
#include <iostream>

using namespace std;

int main() {
    int i = 1;

    // while loop from 1 to 5
    while (i <= 5) {
        cout << i << " ";
        ++i;
    }

    return 0;
}
```

```
#include <iostream>

using namespace std;

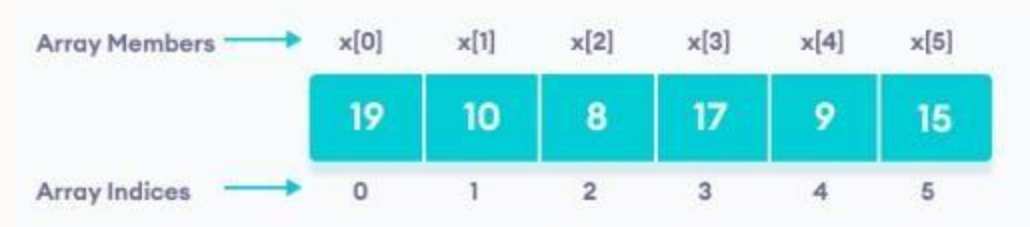
int main() {
    int i = 1;

    // do...while loop from 1 to 5
    do {
        cout << i << " ";
        ++i;
    }
    while (i <= 5);

    return 0;
}
```

ARRAY

An array is a variable that can store multiple values of the same type. It's a homogenous structure.



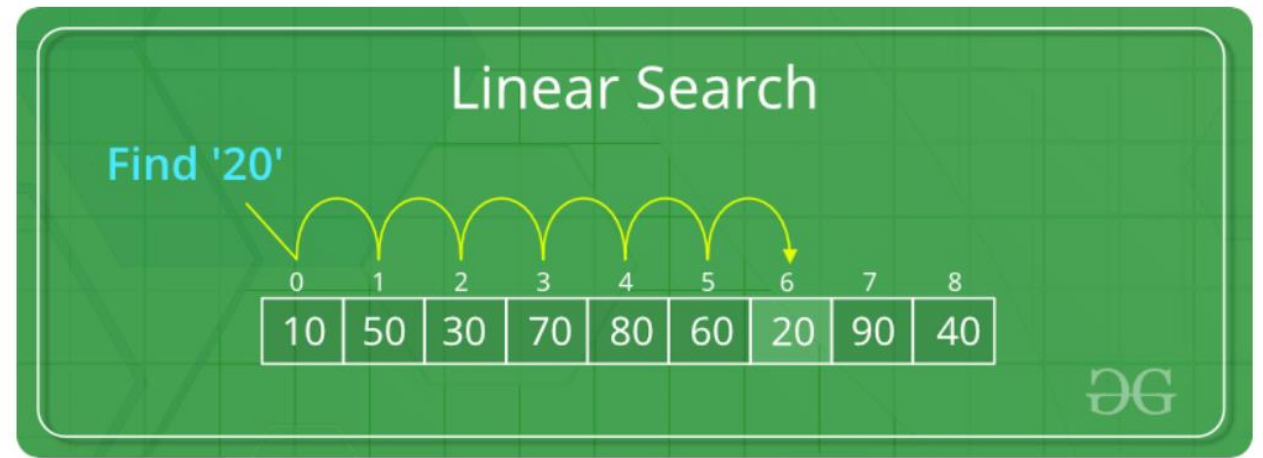
```
int main()
{
    int numbers[5] = {7, 5, 6, 12, 35};
    cout << "The numbers are: ";
    for (const int &n : numbers) { cout << n << " "; }
    cout << "\nThe numbers are: ";
    for (int i = 0; i < 5; i++) {
        cout << numbers[i++] << " " << numbers[++i]; }
    return 0;
}
```

SEARCHING

1. Linear Search
2. Binary Search

Linear Search

- linear search is a search algorithm, also known as sequential search, that is suitable for searching a list of data for a particular value.
- It operates by checking every element of a list one at a time in sequence until a match is found.



```
#include <iostream>
using namespace std;
int main(void)
{
    int arr[] = { 2, 3, 4, 10, 40 };
    int x = 65;
    int n = sizeof(arr) / sizeof(arr[0]);
    int i;
    int found=0;
    bool flag = false;
    for (i = 0; i < n; i++) {
        if (arr[i] == x) {
            found=i;
            flag = true;
        }
    }
    (flag == false)? cout<<"Element is not present in array" : cout<<"Element is present at index "
    <<found;
    return 0;
}
```


SEARCHING

Binary Search

- A binary search is a more efficient algorithm if the data is sorted, in increasing order.
- The strategy of binary search is to check the middle (approximately) item in the list.
- If it is not the target and the target is smaller than the middle item, the target must be in the first half of the list.
- If the target is larger than the middle item, the target must be in the last half of the list.
- Thus, one unsuccessful comparison reduces the number of items left to check by half!
- The search continues by checking the middle item in the remaining half of the list.
- If it's not the target, the search narrows to half of the remaining part of the list that the target could be in.
- The splitting process continues until the target is located or the remaining list consists of only one item. If that item is not the target, then it's not on the list.

```
#include <iostream>
using namespace std;
int main()
{
int arr[] = {1, 3, 7, 15, 18, 20, 25, 33, 36, 40};
int count = sizeof(arr)/ sizeof(arr[0]);
int num = 7;
int first = 0;
int last = count-1;
while (first <= last) {
    int middle = (first+last)/2;
    if(arr[middle] < num){
        first = middle + 1;
    }
    else if(arr[middle] == num)
    {
        cout<<num<<" found";
    }
    else
    {
        last = middle - 1;
    }
}
return 0;
}
```

Sorting

Bubble Sort

```
#include <iostream>
using namespace std;
int main()
{
    int array[] = {1, 3, 7, 15, 18, 20, 25, 33, 36, 40};
    int size = sizeof(array)/ sizeof(array[0]);
    for(int i = 0; i<size; i++)
    {
        for(int j = 0; j<size-i-1; j++)
        {
            if(array[j] > array[j+1])
            {
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}
```

```
cout<<"Sorted Array: ";
for(int i = 0; i<size; i++)
{
    cout<<array[i]<<" ";
}
return 0;
}
```