OBJECT ORIENTED CONCEPT AND PROGRAMMING UNIT -1

-Madhavi Dave

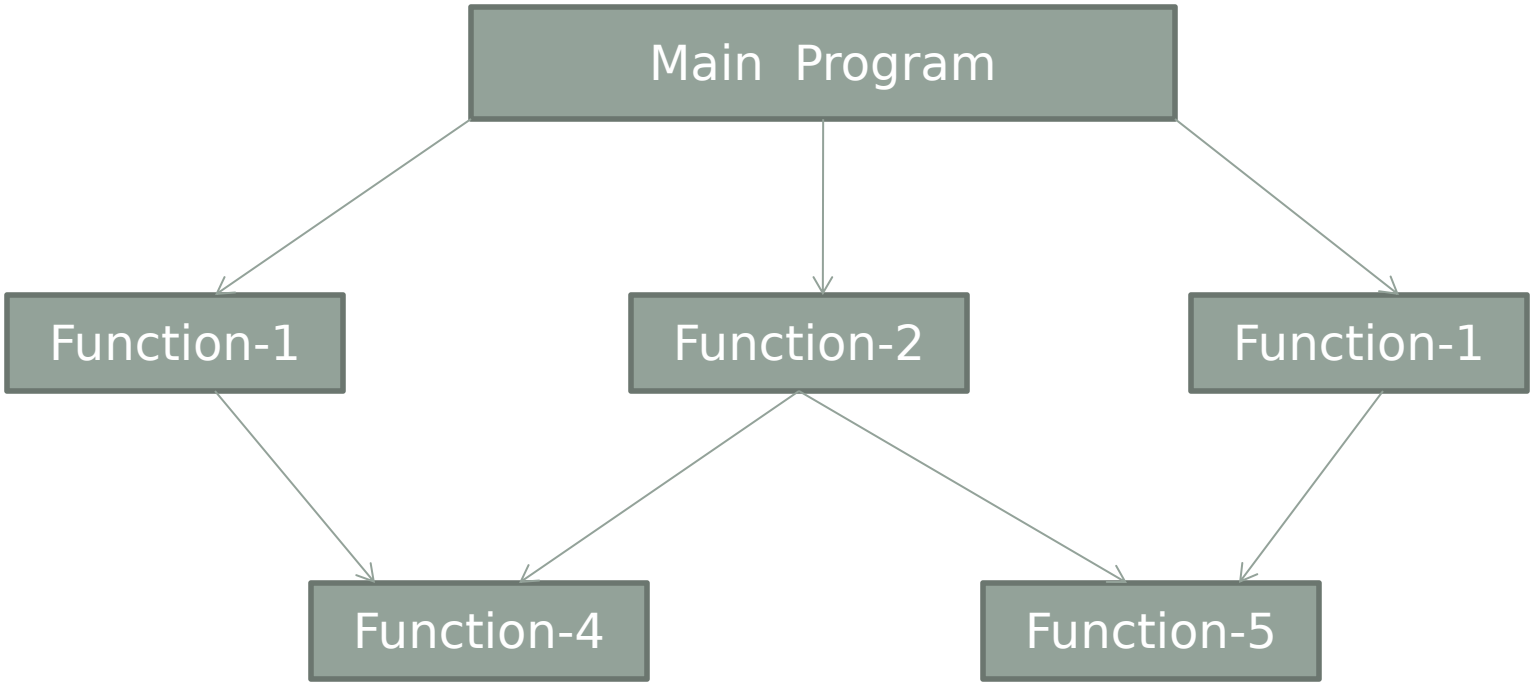
PRINCIPLES OF OBJECT-ORIENTED PROGRAMMING

Procedure Oriented Programming

- High level language such as COBOL, C, FORTRAN is known as procedure-oriented programming.

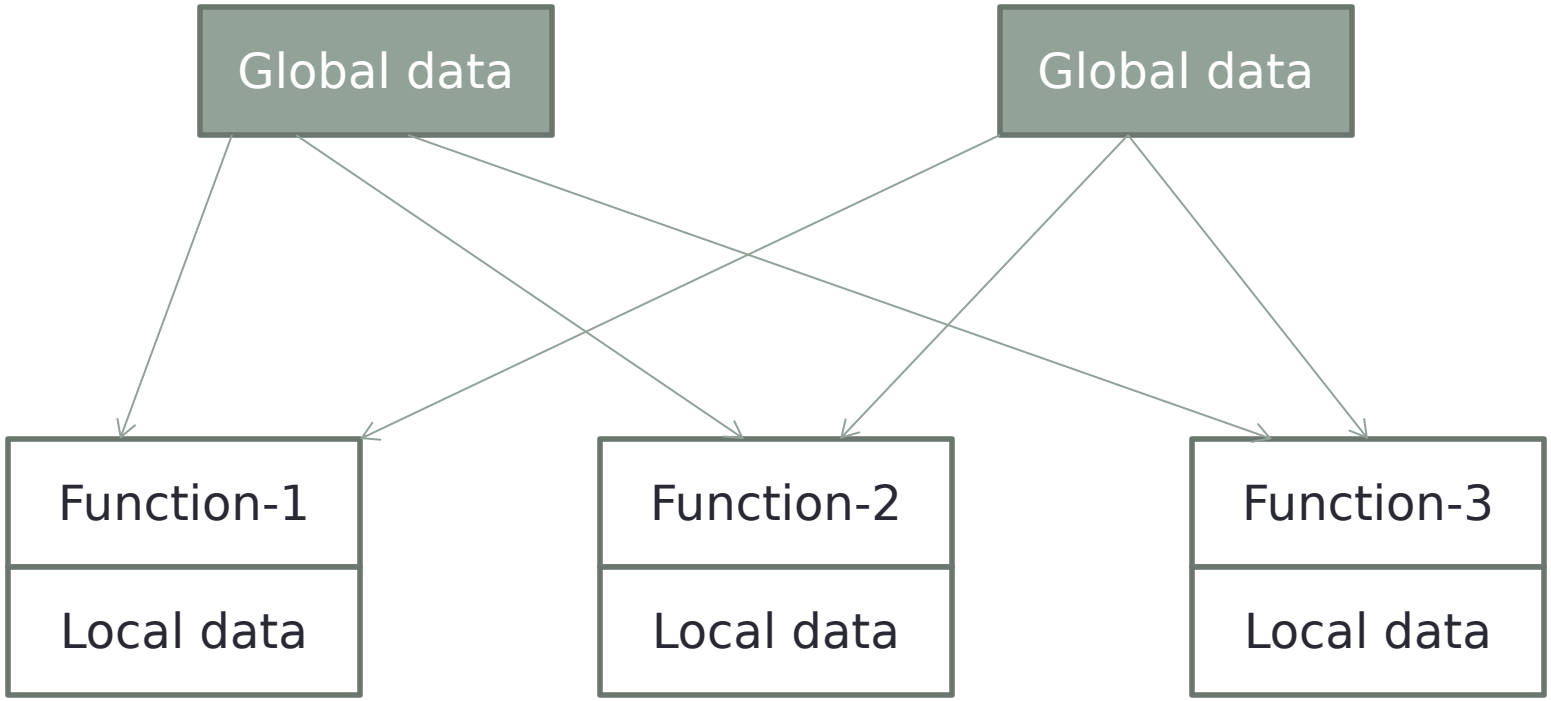
- In POP approach , the problem is viewed as a sequence. like reading, calculating and printing.

Cont…



Structure of procedure oriented programs

Cont..



Relationship of data and function in procedural programming

Cont..

- Characteristics of procedure-oriented programming are:

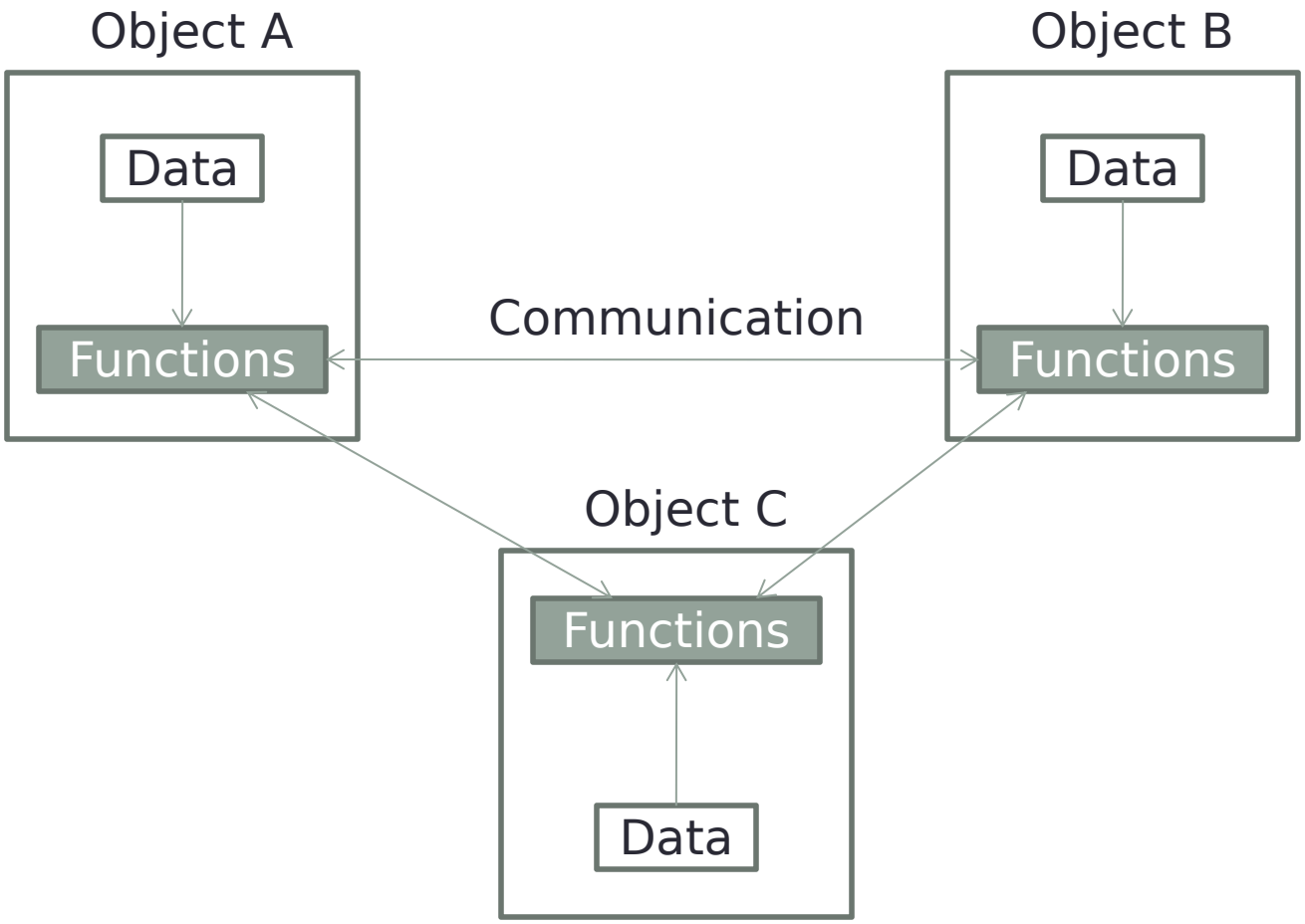
1. Emphasis is on doing things(algorithms)
2. Large program are divided into programs called functions
3. Most of the function shares the global data.
4. Data move openly around the system function to function.
5. Function transfer data from one form to another.

Object-Oriented Programming

- The major motivating factor in oop is to remove some of the flaws encountered in the procedural approach.

- OOP allows decomposition of a problem into a number of entities called *object* and then build data and function around these objects.

Cont…


Object A
Data
Functions
Communication
Object B
Data
Functions
Object C
Functions
Data

Cont…

- Features of object-oriented programming are:

1. Emphases on data rather than procedure
2. Programs are divided into what are known as objects
3. Data structure are designed such that they characterize the objects.
4. Function that operates on the data of an object are tied together in the data structure.
5. Data is hidden and can not be access by external function.
6. Object may communicate with each other through functions.
7. New data and function can easily added whenever necessary.
8. Follows *bottom-up* in program design.

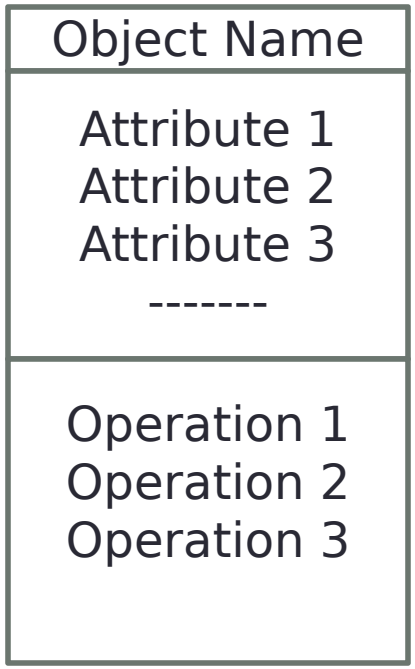
Basic concept of OOP

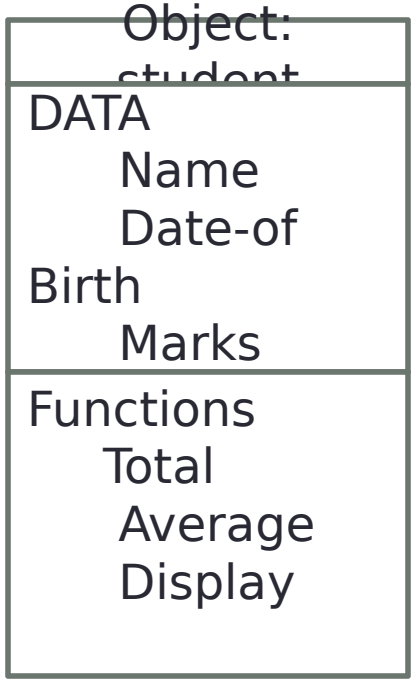
- Object
- Class
- Encapsulation and data abstraction
- Inheritance
- Polymorphism
- Dynamic binding
- Message passing
- Delegation

Objects

- It is the basic run time entities in an object-oriented system
- They may represent a person, a place, a bank account....
- Program object should be chosen in such a way that they match closely with the real world object.
- Object is an instance of class.

Cont..





Way to represent an object

Example

Classes

- The entire set of data and code of an object can be made a user defined data type with the help of a class.
- In fact, object are the variables of the type *class.*
- *Thus class is a collection of objects of similar types.*
- classes are user defined data type.
- Syntax to create object is:
  - Class_name object_name

Encapsulation and data abstraction

- The wrapping of data and function into single unit is known as *encapsulation.*
- Data is not accessible outside world, only function which are wrapped in the class can only access it.
- This insulation of data from direct access by the program is called *data hiding* OR *information hiding.*

Cont..

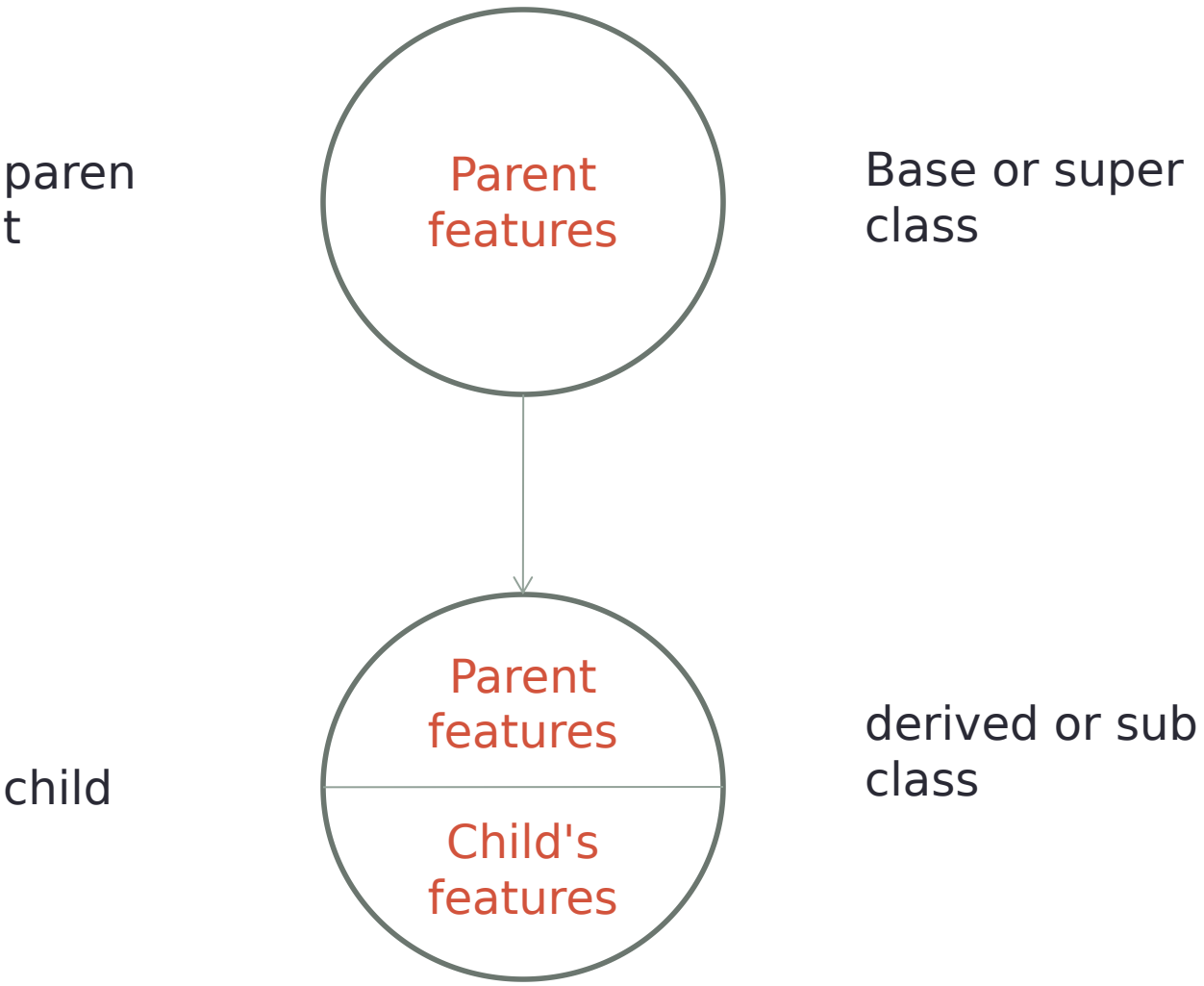
- Abstraction refers to the act of representing essential features without including the background details or explanations.
- Classes use the concept of abstraction, they are known as Abstract data Types(ADT).
- The attributes called data members and function called member function.

Inheritance

- It is a process, by which one object can acquire the properties of another object.
- It allows the declaration and implementation of one class to be based on an existing class.(reuse).
- It can be also define as the mechanism that permits a class to share he attributes and operations defined in one or more classes.

Cont…

- Example:



Cont..

- Single inheritance
- Multiple inheritance
- Multilevel inheritance
- Hierarchical inheritance
- Hybrid inheritance

polymorphism

- It is Greek term, means the ability to take more than one form.
- An operation may take different behaviors in different instance.
- The behavior depend upon the types of data used in the operation.
- For ex. Consider the operation for addition.

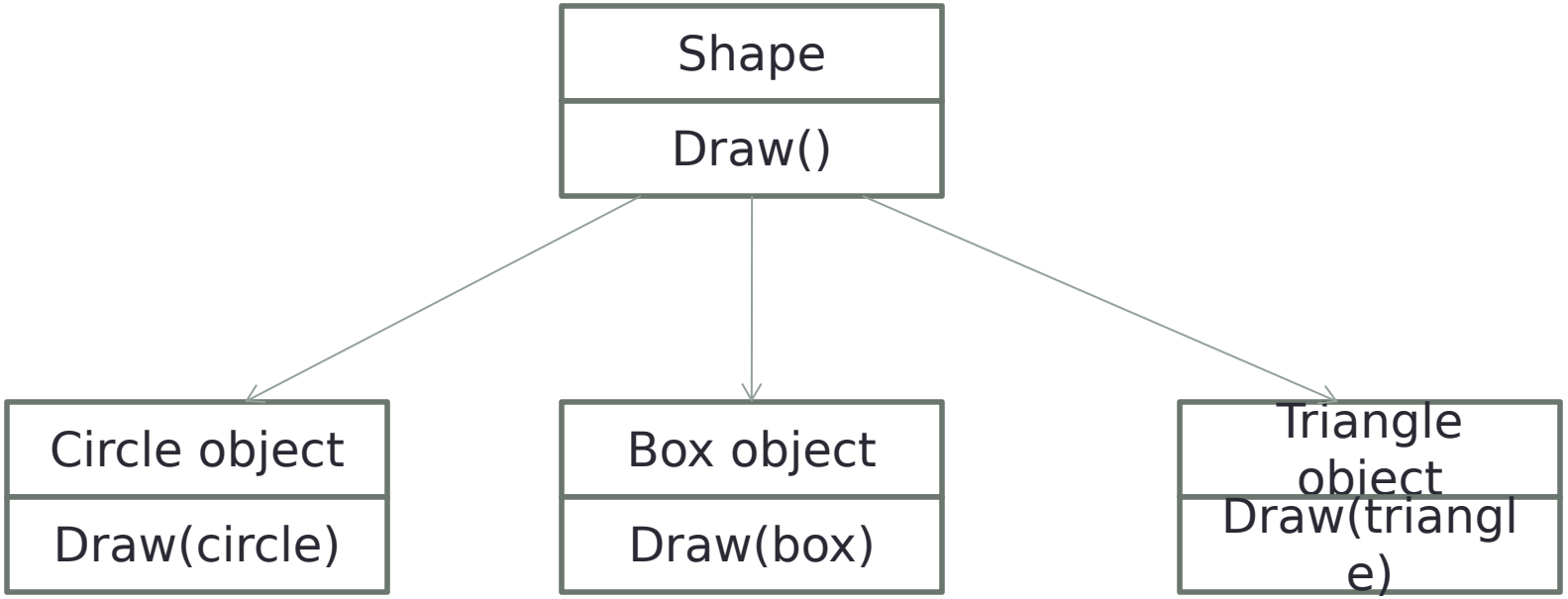
  if two numbers, then addition operation will generate sum.

  if the operands are strings then it perform concatenation.

  It is operation overloading.

Cont..

- Using single function name to perform different types of task is known as function overloading.



Dynamic Binding

- Binding means the linking of a procedure call to the code to be executed in response to the call.
- Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time.
- It is associated with polymorphism and inheritance.

C++ TOKEN, EXPRESSION AND CONTROL STRUCTURES

Tokens

- The smallest individual units in a program are known as tokens. C++ has following tokens :
- Keywords
- Identifiers
- Constants
- Strings
- Operators

A C++ program is written using these tokens, spaces and syntax of the language.

Keywords

- Keywords are the explicitly reserved words for the language and cannot be used as names for the program variables or other user-defined program elements.
- Eg. Auto, break, case, const, class etc.

Identifiers and Constants

- Identifiers means the names of the variables, functions, arrays, classes etc. used in your program.
- Each language has their own rules for naming these identifiers.
- Some rules for both c and c++ are

- i) Only alphabetic characters, digits and underscores are permitted.
- ii) The name cannot start with a digit.
- iii) Uppercase and lowercase letters are different
- iv) A declared keyword cannot be used as a variable name

Constants

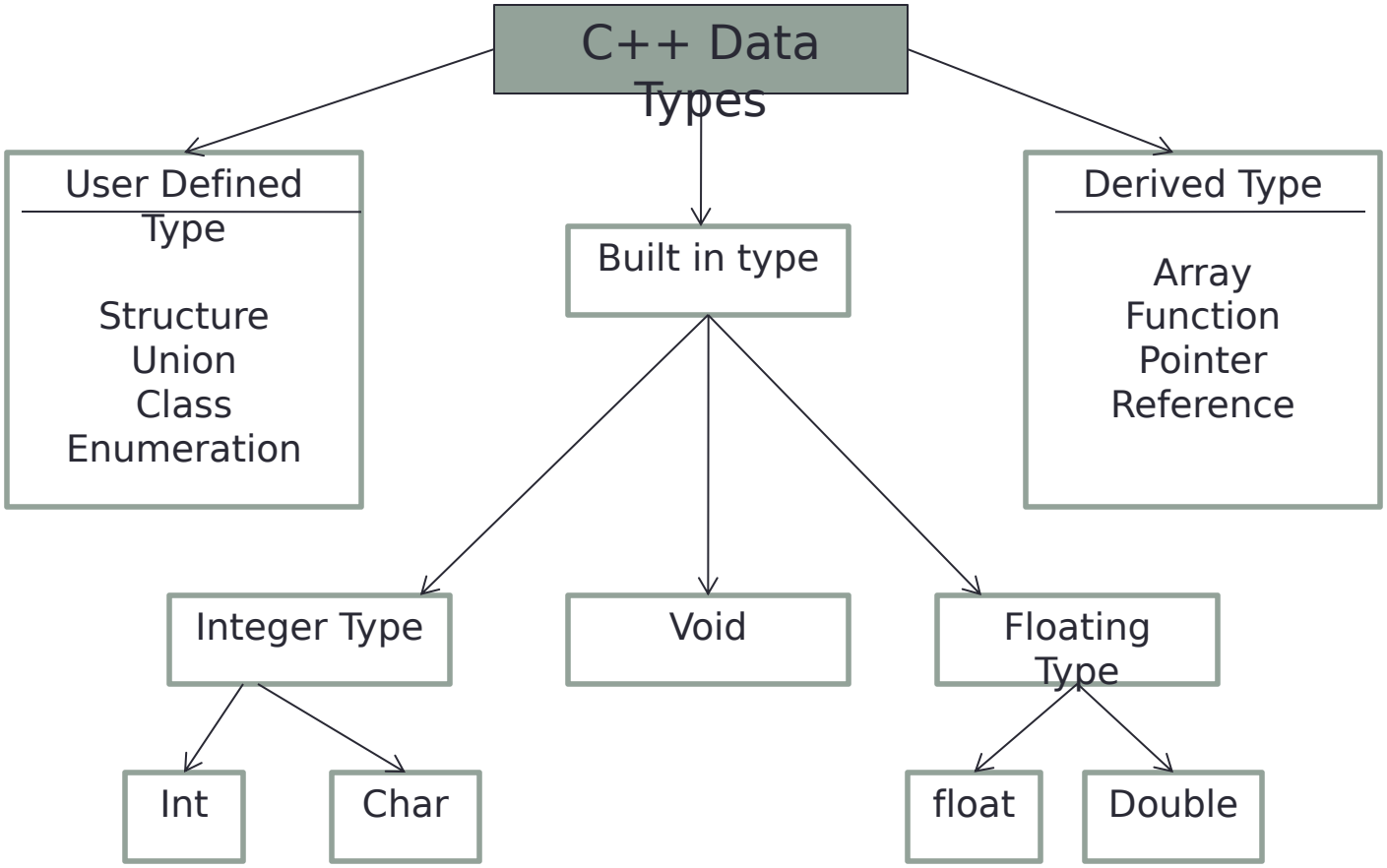
- Constants means the fixed values that do not change during the execution of a program.
- They are integers, characters, floating nos. and strings
- Eg.

123 -- int constant

“C++ ” -- string constant

‘A’ -- Character constant

Basic Data Types



Reference Variable

- A reference variable provides an alias or alternative names for a previously defined variable.
- Eg. We can make a variable sum a reference to the variable total and then use sum and total interchangeably.
- Syntax : datatype & refName = variable name
- Eg float total = 100;
  - float & sum = total

Operators in C++

- :: - Scope resolution operator
- ::* - Pointer-to-member declarator
- ->* - Pointer-to-member operator
- .* - Pointer-to-member operator
- delete - Memory release operator
- Endl – Link feed operator
- New – Memory allocation operator
- Setw – Field width operator

C++ control structures

- **Selection**
  - **if**
  - **if . . . else**
  - **switch**
- **Repetition**
  - **for loop**
  - **while loop**
  - **do . . . while loop**

CONTROL STRUCTURES

Use logical expressions which may include:

6 Relational Operators

< <= > >=

== !=

3 Logical Operators

! && ||

Operator	Meaning	Associativity
!	NOT	Right
*, / , %	Multiplication, Division, Modulus	Left
+ , -	Addition, Subtraction	Left
<	Less than	Left
<=	Less than or equal to	Left
ı	Greater than	Left
ı=	Greater than or equal to	Left
==	Is equal to	Left
!=	Is not equal to	Left

Example

```
int    Number;
float  X;

( Number != 0) && ( X < 1 / Number )
```

Conditional statements

- Syntax

  if (*expression*)

  statement1

  else

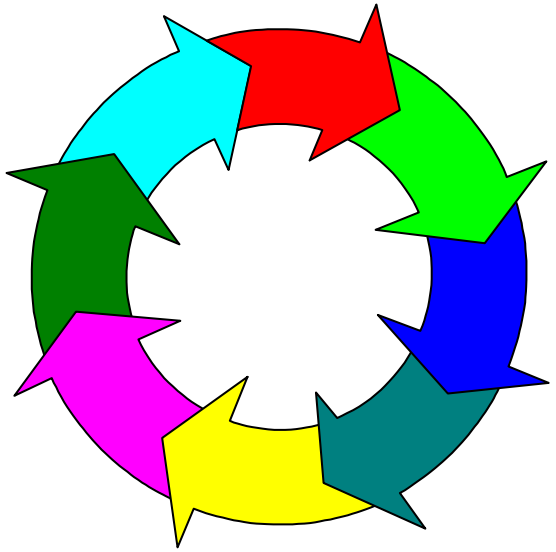
  statement2

else clause is optional

Iteration statements

- while-statement syntax

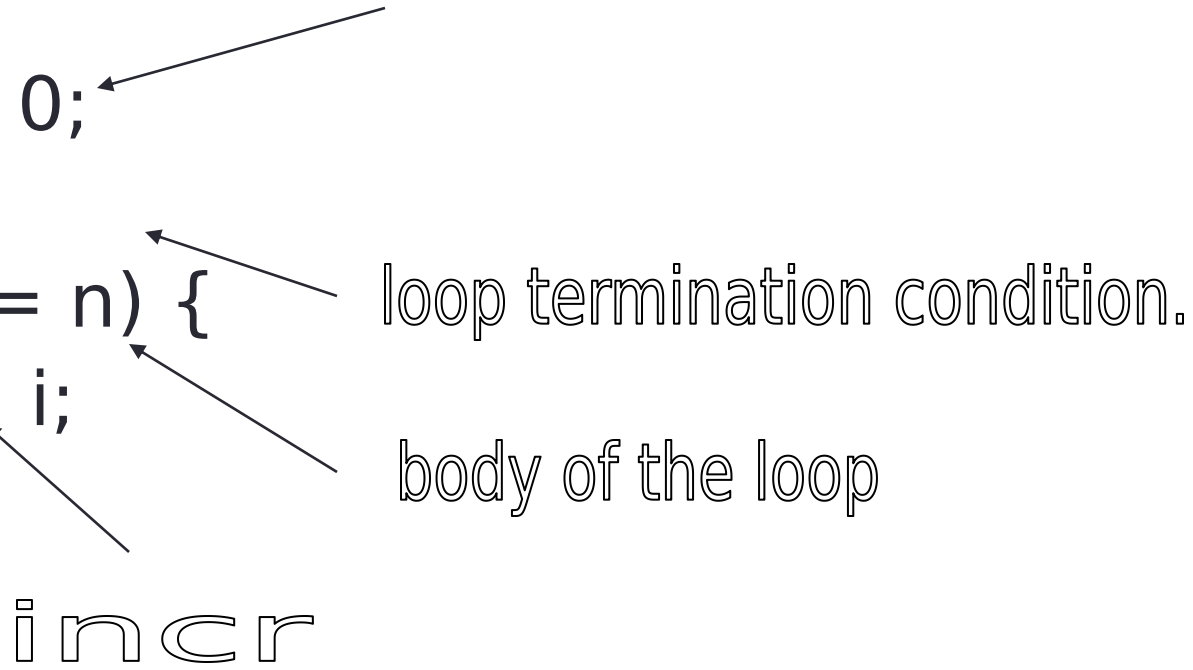
  while (*expression*)
  statement



It's a pre-test loop.

Iteration statements

```
// compute sum = 1 + 2 + ... + n
// using a while loop
int i;
int sum = 0;
i = 1;
while (i <= n) {
   sum += i;
   i++;
}
```



Iteration statements

```
// compute sum = 1 + 2 + ... + n
// using for loop

int sum = 0;
for (int i = 1; i <= n; ++i) {
   sum += i;
}
```

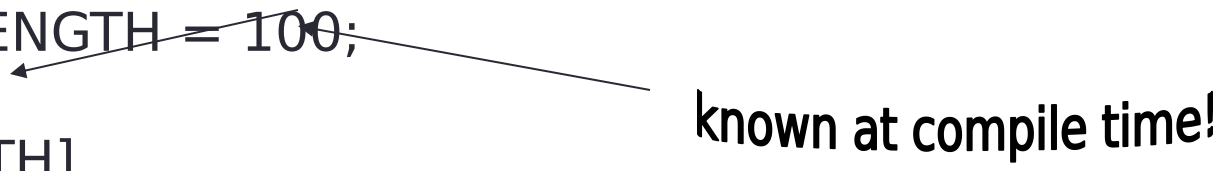
Switch

```
switch (letter) {
   case ‘N’: cout < “New York\n”;
             break;
   case ‘L’: cout < “London\n”;
             break;
   case ‘A’: cout < “Amsterdam\n”;
             break;
   default:  cout < “Somewhere else\n”;
             break;
}
```

Simple arrays

- subscripts can be an integer expression
- In the declaration, the dimension must be a constant expression

```
const int LENGTH = 100;
...
int a[LENGTH]
...
for (int i=0; i<LENGTH; i++)
   a[i] = 0;  // initialize array
```



Functions:
3 parameter transmission modes

- pass by *value (default)*
- pass by *reference (&)*

local copy

pass the address

- pass by *const reference (const &)*

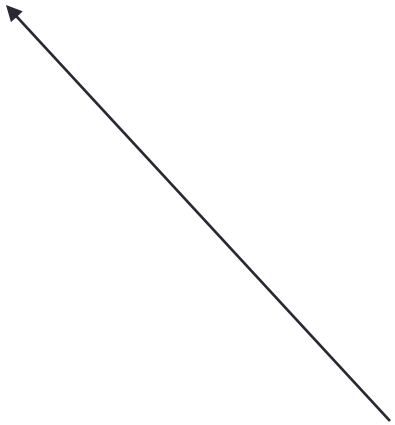
good for big structures

Functions: example of pass by value

```
int sqr(int x) {



}
```



The compiler makes a local copy!

The Swap Function

```
swap (a, b);

void swap(int x, int y)
{
  // Create a temporary variable
  int temp;
  temp = x;
  x = y;
  y = temp;
}
```

Passing values by reference!

- C/C++ passes parameters by value, i.e. a copy of the variable is passed to the function, not the actual value itself.
- C++ can pass the actual variables themselves - known as *passing parameters by reference*.
- To pass a parameter by reference we place & between the parameters type name and the parameter tag.

The New Swap Function

```
void swap(int& x, int& y)
{
  // Create a temporary variable
  int temp;

  temp = x;
  x = y;
  y = temp;
}
```

Pointers

- Pointers in C++ are same like in C, but some characteristics are different from C.
- Void Pointer :- “pointer to void” in c++, it points to a value which does not have a type such as int, float etc.
- Eg. Int* F1

```
        void * S1
 S1 = F1 //valid in c and C++
 F1 = S1 // not valid in C++
 For c++ we need to convert it in int type as c++ is strictly
   typed language
 F1 = (int*) S1;
```

- Constant Pointer :-It is one that cannot point to anything other than what it is pointing to at the time of its definition. This means the address cannot be changed but the content of the address can be changed.

- Eg int * const S1 = &content1
- S1++ // not possible

- Pointer to Constant :- A pointer to Constant can point to any memory location, but the content to which it points cannot be changed.
- int const* S1 = &Content;
- *S1 = 100 // not allowed.

Functions

- Diving a program into Functions is one of the major principles of top-down, structured programming.
- It also reduce the size of the program by calling and using them at different places in program
- Function Prototyping :- It is a declaration statement in the calling program.

  type function-name (argument-list);

- Call by Reference :- Reference variables in c++ permits us to pass parameters to the functions by reference. This means ‘formal’ arguments in the called function become aliases to the ‘actual’ arguments in the calling function.

- Inline Functions:- To eliminate the cost of calls to small functions C++ has Inline functions. That means a function is expanded in line when it is invoked.
- That means the complier replaces the function call with the corresponding function code
- Syntax: inline function-header

```
{
  function-body
}
```

Eg

```
inline double square(double r)
{       return r * r;
}
c = square(3);
```