C++ Programming Notes

Contributor: Abhishek

[TutorialsDuniya.com]

Computer Science Notes

Page No.
Date

The first widely used high level lang. FORTRAN [Formula Translation] was created by John Backus & IBM team in 1957.

After FORTRAN, COBOL [Common Business Oriented Lang.] was developed.

Other lang. BASIC, Pascal, Ada and C.

Pseudocode: A precise algorithmic description of program logic.

Blackbox testing assumes that tester knows nothing about program,
Whitebox testing assumes that tester knows everything about program.

In 1960, ALGOL [ALGOrithmic Language] was first lang. to use block str. C++ is derived from ALGOL.

In 1967, Martin Richards invented BCPL [Basic Combined Programming Lang.] which was a typeless lang.

In 1970, Ken Thompson invented typeless system programming lang. B.
B was used to develop first version of UNIX.

Page No.
Date

In 1972, working at Bell Laboratories, Dennis Ritchie designes C, a comb. of BCPL and B but with lotsa type.

In 1980, Bjarne Stroustrup extended C to include classes.

In 1985, after Stroustrup added virtual fn & overloading, new lang. become known as C++.

One, and only only one, of fn in a program must be named 'main'.

Main is the starting point for the program.
Main and other fn in a program contain 2 types of code: definition & Statements.

* Definition describes data that you will be using in fn. Definition in a fn are known as local definition because they are visible only to fn that contain them.
* Statements are inst. to comp. that cause it to do something, such as add 2 numbers.

It is almost impossible to write even smallest of programs without atleast one system library.

A namespace is a collection of names belonging to a group.

The namespace for standard system libraries is standard (std).

Page No. ____ TAYAL
Date

Structure of a C++ program

Preprocessor Directives

Global declaration

```
int main()
{   Local definition
    Statements
}

int fun(...)        // user def. func"
{   Local def.
    Statement
}
```

C Library	C++ Library	Usage
ctype.h	cctype	char type library
float.h	cfloat	floating point char library
limits.h	climits	Integer value library
math.h	cmath	Mathematical fn library
stdlib.h	cstdlib	Standard library
string.h	cstring	string library

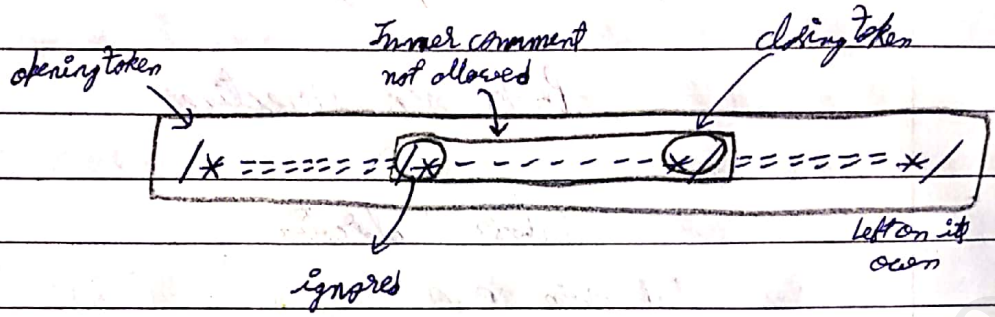
Comments are used wherever it is necessary to explain a point about the code.

Single line comment : // 1 line comment

Multiple line comment : /* This is
2 line comment */

Page No. ——— TAYAL
Date

Comments cannot be nested



Identifiers allows us to name data and other objects in the program.
Each piece of data in comp. is stored at a unique address.

① First character must be alphabetic char or underscore.
② Identifier must consist of alphabetic characters, digits & underscore.
③ Identifier cannot duplicate a reserved words.

Valid names	Invalid names
a , a1	$sum // $ is illegal
Abhi_Sharma , AbhiSharma	2names // can't start with 2
_AKS , _Abhi9101	Abhi Sharma // no spaces
pi , PI	int // reserved word

Keywords / Reserved words
int, char, float, void, for, double etc.

A type defines a set of values & a set of operations that can be applied on those values.

Page No. ____ TAYAL
Date

C++ contains 5 standard types

① Void
② int (short for integer)
③ char (short for character)
④ float (short for floating point)
⑤ bool (short for boolean)

1. Void

The void type has no values & no operations.

2. Integer

An integer type is a number without a fractional part.

Type	Sign	Byte	Min. value	Max. value
short int	signed	2	-32,768	32,767
	unsigned		0	65,535
int	signed	2	-32,768	32,767
	unsigned		0	65,535
long int	signed	4	-2,147,483,648	2,147,483,647
	unsigned		0	4,294,967,295

C++ provides an operator, sizeof, that will tell you exact size of any data type.

sizeof(short int) <= sizeof(int) <= sizeof(long int)

If integer is signed, then one bit must be used for sign (0 if plus, 1 is minus).

Page No. ____ TAYAL
Date | | . | |

The unsigned integer can store a +ve number that is twice as large as signed integer of same size.

C++ has a library, <climits>, that contains size information about integers.

3. CHAR

A character in C++ can be implemented as a small integer (b/w 0 and 255). For this reason, C++ often treats a char like an integer.

4. Floating point

A floating point type is a number with a fractional part, such as 41.786.

Type	Bytes
float	4
double	8
long double	10

sizeof (float) <= sizeof (double) <= sizeof (long double)

5. Bool

C++ implementation of logical data is bool type.

In C++, Boolean constants are true & false.

any nonzero no. is considered true & zero is false.

Page No. TAYAL
Date

Variables are named memory locations that have a type and a size which is inherited from their type.

Each variable in a program must be declared & defined. In C++, a declaration is used to name an object, such as variable.
Definitions are used to create the object.

A variable's type can be any of the data types, such as char, int and float. The one exception to this rule is type void; a variable cannot be type void.
To create a variable, first specify the type & then its identifier.

int (variable's type) Abhishek (variable's identifier);

Constants are data values that cannot be changed during execution of a program.

There is no way to specify short int const.

Char const. are enclosed b/w 2 single quotes.

The back slash (\) is known as escape character. It is used when char. we need to represent that not have any graphic associated with it; when it cannot be printed or when it cannot be entered from keyboard.

Page No. TAYAL
Date

A string const is a sequence of zero or more characters enclosed in double quotes.
An empty string is simply 2 double quotes in succession.

```
""                      // null string
"'Good Morning'"        // 'Good Morning'
"\"Good Morning\""      // "Good Morning"
'\0'                    // null char.
'\\'                    // backslash (\)
```

A null char represents no value.
As a char, it is eight 0 bits.
An empty string is a string containing nothing.

'\0' → Null char
"" → Empty string

Coding Constants

① Literal Const.

A literal is an unnamed const. used to specify data.

```
'A'         // char literal
5           // numeric literal 5
3.14        // float literal
"Abhishek"  // string literal
```

② Defined constants

```
#define Abhishek 1.786
```

Page No. TAYAL
Date

(3) Memory const.

```
const int Abhi = 9101;
const float pi = 3.14;
```

Memory const. are more preferred.

① They conserve memory because they are declared & stored only once. While defined const. are actually literals and may be stored wherever they are referenced.

② Programs with memory const. are easier to read. When defined const are used, the precompiler substitutes the defined const. value wherever it is used, but the program listing shows the original code.

Data Sources

C++ defines one standard source, standard input, and 2 standard destinations, standard output & standard error.

They are called standard because C++ creates them automatically when we need to receive data from console keyboard or send data to console monitor.

By default, standard input is associated with console monitor.

Keyboards and monitors handle data only as a sequence of characters.

Standard Streams

A stream is an abstract representation of an input data source or output data destination

Page No. Date TAYAL

C++ defines 4 standard streams :—

1. Console input [associated with standard input (keyword)]
2. Console output [associated with standard output (monitor)]
3. Console error
4. Console log

A buffer is a temporary storage area that holds data while it is being received or accumulated.

Console log is buffered while console error is unbuffered.
The console log output is held until the system determines that it should be written. While console error output is displayed on console immediately after it is written.

Writing errors
cerr << "A buffered error message \n";
clog << "An unbuffered error message \n";

Manipulators
Manipulator fn format output so that it is presented in a more readable fashion for the user.

Those without parameter argument are found in the base input/output stream library (<iostream>)

Page No. ——— TAYAL
Date

Those with prometer arguments, such as set width, set precision and set fill are found in a specid library known as input/output manipulator library (<iomanip>).

Manipulator	Scope	Use
endl	one time	New line
dec	permanent	formats output as decimal
oct	permanent	formats output as octal
hex	permanent	formats output as hexadecimal
fixed	permanent	Sets float-point decimals
showpoint	permanent	Shows decimal in floating point values
setw(...)	one time	Sets width of output field
setprecision(...)	permanent	Specifies no. of decimals for floating point
setfill(...)	permanent	Specifies fill character

There are 3 general manipulators :-
① New line ② Set width ③ Set fill
These are general manipulators because they can be used with all types of data.

① Newline (endl)
Newline manipulator terminates current line & starts a new one. It has same effect as newline ('\n').

```
cout << "Abhishek \n" << "Sharma";
cout << "Abhishek" << endl << "Sharma";
```

output: Abhishek
Sharma

Page No. TAYAL
Date

2. Set Width (setw)

Set width allows us to set minimum width for an output field.

Set width sets the minimum space, if data need more space in output, cout will override set width request & use whatever space is required.

Right justification places data in an output area with data oriented to right & fill characters on the left. The fill characters is generally spaces, although we can control the actual character used.

Left justification starts the data on left of an output area and adds trailing fill characters on right.

```
cout << setw(5) << "Abhishek";      Abhishek
cout << setw(12) << "Abhishek";     ---Abhishek
```

Set Fill (setfill)

When the width of a print area is larger than data values to be placed in it, C++ uses a fill character for non data area.

The default fill character is space.

* float a = 9101.786;

```
cout << setw(10) << a;                          9101.786
cout << setw(10) << setfill('*') << a;          **9101.786
cout << setw(10) << a;                          **9101.786
cout << setw(10) << setfill(' ') << a;          9101.786
```

Page No. TAYAL
Date

The fill character is a permanent-until-changed manipulator. To change back to a space fill, it must be reexecuted.

Integer Manipulators

Integer manipulators are used to change the display format for integer values.
Decimal manipulator is default.
dec tells cout to print value in decimal.
octal (oct) tells cout to print value using octal number system.
hexadecimal (hex) tells cout to print in hexadecimal.
Each of these manipulators sets the printing until it is reset by another manipulator.

```
int a = 123;
cout << setw(5) << a;                    123
cout << hex << setw(5) << a;             7b
cout << oct << setw(5) << a;             173
cout << dec << setw(5) << a;             123
```

Floating point manipulator

There are 3 floating point manipulators :-

① Fixed

Fixed manipulator tells cout that floating-point no. are to be displayed with fixed point rather than floating point no.
Floating point no are stored in memory in 2 parts : a mantissa & an exponent.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

When no. to be displayed is very large or very small, the fixed manipulator displays 2 parts seperated by exponent Taken.

1.2345678e+06

its fixed format : 1234567.875

2. Set precision

Set precision is used to control the no. of decimal places to be displayed.

3. Show point

When we use zero precision with a floating point no., C++ print it without a decimal point, which makes it look like an int. To display value with a decimal point, we use showpoint.

```
float a=1.0, b=1.234, c=1234.5678;
cout << a << b << c;
cout << fixed;
cout << a << b << c;
cout << setprecision(2);
cout << a << b << c;
cout << setprecision(0);
cout << b;
cout << setprecision(0) << showpoint;
cout << b;
```

```
1.0
1.234
1234.5678 1.234e+04
1.000000
1.234000
1234.567800
1.00
1.23
1234.57
1
1.0
```

Page No. ____ TAYAL
Date

★ The fn header for main should be complete.
(a) If you forget the parenthesis after main, you will get a compile error.
(b) If you put a semicolon after the parenthesis, you will get a compile error.
(c) If you misspell main you will not get a compile error but you will get an error when you try to link your program.
All programs must have a fn named main.

Not terminating a comment block with a close token (*/) is a compile error.
Not including req. libraries such as <iostream>, at beginning of your program is linker error.
If you misspell the name of a fn, you will get an error when you link the program.
For ex: if you misspell cin or cout, your program will compile without errors, but you will get a linker error.

★ Expression
An expression is a sequence of operands & operators that reduces to single ~~variable~~ value.
Expression always reduces to single value.

For ex: 2+5 is an exp. whose value is 10.

The value can be any type other than void.

Page No. ——— TAYAL
Date | | |

An operator is a lang-specific syntactical token that requires an action to be taken.

An operand receives an operator's action.

Assignment expression
The assignment exp. has a value as a result.
The value of total exp. is value of exp. on right of assignment operator (=).
The result ~~[illegible]~~ places the exp. value in operator on left of assignment operator.

Simple assignment
a=5 , b=a+1 , i=i+1

Compound assignment
x+=y , x*=y , x/=y , x%=y

Precedence
It is used to determine the order in which diff operators in a complex expression are evaluated.

Associativity
It is used to determine the order in which operators with same precedence are evaluated in a complex expression.

Page No.
Date

Left Associativity

3 * 8 / 4 % 4 * 5

((((3*8)/4)%4)*5) = 10

Right Associativity

There are only 3 types of exp. that associate from right : 1. unary exp.
2. Conditional ternary exp.
3. Assignment exp.

a += b *= c -= 5

(a = 3 + (b = (5 * (c = 8 - 5))))

hence,

a = 18, b = 15, c = 3

Side Effects

A side effect is an ~~act~~ action that results from evaluation of a exp.

x = x + 4

Assume that x has an initial value of 3, value of exp on right of assignment operator has a value of 7. Whole exp. also has a value of 7. And as a side effect, x receives value 7.

Type of side effect	exp. type	Example
Pre effect	unary prefix inc.	++a
Pre effect	unary prefix dec.	--a
Pre effect	func. call	doIt(....)
Pre effect	assignment	a=1 a+=x
Post effect	postfix inc	a++
Post effect	postfix dec.	a--

Page No. ____ TAYAL
Date

--a * (3+b) / 2 - c++ * b

a = 3, b = 4, c = 5.

1. Calc value of parenthesized exp. (3+b)

--a * 7 / 2 - c++ * b

2. Evaluate the c++ postfix exp.

--a * 7 / 2 - 5 * b

3. Evaluate the --a prefix exp.

2 * 7 / 2 - 5 * b

4. Multiplication & division are now evaluated using associativity rule, left to right.

14 / 2 - 5 * b ⇒ 7 - 5 * 4 ⇒ 7 - 20

5. Last step is to evaluate subtraction.

7 - 20 ⇒ -13

After side effect variables are; a = 2, b = 4, c = 6

☆ Warning

In C++, if a single variable is modified more than once in an exp., result is undefined.

Never use a variable affected by side effect more than once in an exp.

Page No.
Date

Implicit type Conversion

In this, the compiler automatically converts the value with lower precision to value with higher precision.

Only the value is converted, the value & type of memory variable are unchanged.

Explicit type conversion

In this, user itself converts the value with lower precision to value with higher precision.

Static cast

Syntax : static_cast<type> (exp)

average = static_cast<float> (total score)/numscore;

C-style cast

(type) exp.

```
int Num1 = 100; Num2 = 45; double fltNum3;
fltNum3 = static_cast<double> (Num1 / Num2);
cout << fltNum3;
fltNum3 = static_cast<double> (Num1) / Num2;
cout << fltNum3;
```

Result : 2
2.222222

Page No. ____ TAYAL
Date

An exp. statement is terminated with a semicolon. The semicolon is a terminator & it tells compiler that statement is finished.

Compound Statement (block)
All C++ fn contains a comp. statement known as fn body.

A comp. statement consists of an opening brace, optional declaration, definition & statements, followed by closing brace.

* A comp. statement does not need a semicolon. If you put a semicolon after a closing brace, compiler think that you have put an extra null statement after the comp. statement.

* Define const. is an automatic substitution. one common mistake is to place a semicolon at end of command.

```
#define SALES_TAX 0.1234;
---
salestax = SALES_TAX * salesAmt;
```

After substitution would be following erroneous code because a semicolon has been coded after const. value.

```
salestax = 0.1234; * salesAmt;
```

Page No. TAYAL
Date

It is a compile error to use a variable that has not been defined.

It is a compile error to forget semicolon at end of an exp. statement.

It is a compile error to terminate a defined const. (#define) with a semicolon.

It is a compile error when operand on left of assignment operator is not a variable.

Ex= (a+3) = b*c;

It is a compile error to use increment or decrement operators with any exp. other than variable identifier.

Ex= (a+3)++

It is a compile error to use modulus operator (%) with anything other than integers.

It is a logic error to use a variable before it has been assigned a value.

It is a logic error to modify a variable in an exp. when variable appears more than once.

Ex= a++ * (a+b)

a= a++ +b

Page No. TAYAL
Date

☆ A fn in C++ can have a value, a side effect or both.

The side effect occurs before value is returned.
Fn value is value of exp. in return statement.

Advantages of fn.

1. Problems can be factored into understandable & manageable steps.

2. fn provide a way to reuse code that is req. in more than one place in a program.
You can save time & efforts by writing the code once as a fn and then calling it whenever you need to compute average.

3. Protect data.

The name of a fn is used in 3 ways :-
for declaration, in a call & for definition

```
// prototype declaration                    (declaration)
void greeting();                                          (definition)
int main()                                  void greeting()
{ greeting();                               { cout << "Hello";
  return 0;                                   return;
}                                           }
```

(call)

Void fn with No parameter

Page No. TAYAL
Date

```
// prototype declaration                    definition
void printOne (int x);                      void printOne(int x)
int main ()          (declaration)          { cout << x;
{  int a = 5;                                 return;
   printOne (a);  (call)                    }
   return 0;
}
```

Void fn with parameter

void fn cannot be used in an exp. They must be a seperate statement.

fn that returns a value may be used in an exp. or as a seperate statement.

Function definition

fn definition contains code for a fn.
definition is is made up of 2 parts :-
① fn header ② fn body

(fn header)

returntype fn_name (formal parameter list)
{
- - -
- - -
- - -
} // fn body

Page No. TAYAL
Date

Function header
fn header consists of 3 parts:
① return type ② fn name ③ formal parameter list

A semicolon is not used at end of fn def header.

Function body
fn body contains declaration & statements for fn.

The type of exp. in return statement must match return type in fn header.

A local variable is a variable that is defined inside a fn & used without having any role in the communication b/w fn.

Prototype declaration
It consists only of a fn header, they contain no code. Like fn definition headers consists of 3 parts: Return type, Func. name, Formal parameter list.

Formal parameters are variables that are declared in the header of fn definition.
Actual parameters are exp. in calling statement.
Formal & actual parameter must match exactly in type, order & number.
Their names, however, do not need to be same.

Page No. TAYAL
Date

Pass by value

A copy of data is is made and copy is sent to fn. This technique results in parameter being copied to variables in called fn. and also ensures that original data in calling fn cannot be changed accidentally.

```
void fun (int num1);

int main ()
{   int a=5;
    fun (a)
    cout << a << endl;
    return 0;
}

void fun (int x)
{   x = x+3;
    return;
}
```

Pass by Reference

Pass by reference sends address of data rather than copy. In this case, the called fn can change original data in calling fn. Although changing data is necessary, it is one of common source of errors.

```
void exchange (int& num1, int& num2);

int main ()
{   int a,b;
    exchange (a,b);
    cout << a << b;
    return 0;
}

void exchange(int& num1, int& num2)
{   int hold = num1;
    num1 = num2;
    num2 = hold;
    return;
}
```

Page No. TAYAL
Date

Logical Data & operators

A piece of data is called logical if it conveys the idea of true or false.

C++ has 3 logical operators:-
① Not ② And ③ OR

operator	Meaning	precedence
!	not	15
&&	logical and	5
||	logical or	4

Zero ⟷ False
Non Zero ⟷ True

Not operator (!)

The not operator is a unary operator with precedence of 15. It changes true value (non zero) to false (zero) & false value (zero) to true value (one).

AND operator (&&)

It is a binary operator with precedence of 5. The result is true only when both operands are true, it is false in other cases.

Page No.
Date

OR operator (||)

It is a binary operator with precedence of 4. The result is false if both operands are false, it is true in all other cases.

Short Circuit method

It stops the evaluation when it knows for sure what the final result to be.
If first operand of logical AND exp. is false, second half of exp. is not evaluated because it is apparent that result must be false.

With OR exp. if first operand is true, there is no need to evaluate second half of exp. the resulting value is set true immediately.

false && (anything) → false

true || (anything) → true

Relational operators

Less than, Less than or equal, greater than, greater than or equal — have higher priority of (10).

Equal & not equal operators have (9) priority.

★ If Else Statement

An if...else statement is a compound statement used to make decision b/w 2 alternatives.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No. ____ TAYAL
Date

Syntactical Rules for if.....else statements

1. Exp. must be enclosed in parenthesis.
2. No semicolon (;) is needed for an if.....else statement.
3. The exp. can have a side effect.
4. Both True & false statements can be any statements or can be a null statements.
5. Both statement 1 & statement 2 must be one & only one one statement. Multiple statements can be combined into a comp. statement through use of braces.
6. We can swap positions of statement 1 & statement 2 if we use complement of orginal exp.

```
if (exp)
    statement 1
else
    statement 2
```

* Nested if Statements
When an if...else is included within an if...else.

Page No.
Date

Dongling else problem
This problem is created when there is no matching else for every if.

Conditional Expression

exp 3 exp1 : exp2

★ C++ provides 2 fn that allows us to terminate
fn : ① exit ② Abort

Both fn are found in standard lib. (cstdlib).

① Exit
Whereas return terminates a fn, exit terminates program regardless of where in program it is executed. While we use it to terminate program because we detected an error, C++ considers it normal termination.
For this reason, termination is orderly; any pending file stream writes are first completed & all streams are closed.

void exit (int completionstatus)

There is one parameter to exit call, an integer value to be passed to operating system.

Page No.
Date

TAYAL

Abort

Abort fn is used to terminate a program abnormally. It is considered a non orderly termination because output streams are not flushed & they are not closed.

When abort is called, program immediately goes to operating system.

The abort fn has no parameter.

```
void abort();
```

Multiway selection

① Switch statement

Switch statement can be used only when selection cond. reduced to an int. exp.

The 'default' label is a specid form of labeled statement. It is executed whenever none of previous case values matches value in switch exp.

If you do not provide a default, compiler will simply continue with statement after closing brace in switch.

Page No.
Date

```
switch (printFlag)
{ case 1: cout << "do case 1 ";
          doCase1 ();
  case 2: cout << "do case 2";
          doCase2 ();
  default : cout << "do default ";
}
```

There are 3 possibilities depending on value in printFlag. If printFlag is a 1, then all 3 print statements are executed. If printFlag is a 2, then first print statement is skipped & last 2 are executed. Finally if printFlag is neither a 1 nor a 2 then only default is executed. In this case, first 2 print statements would be skipped and only last one would be executed.

```
switch (printFlag)
{ case 1: cout << "do case 1 ";
          doCase1 ();
          break;
  case 2: cout << "do case2 ";
          doCase2 ();
          break;
  default : cout << "do default ";
          doDefault ();
          break;
}
```

Switch with break statements

Page No. TAYAL
Date

Switch statement Rules

1. Control exp. that follows keyword switch must be an integral type.

2. Exp. followed by each case label must be a const. exp.
A const. exp. is an exp. that is evaluated at compilation time, not run time

3. No 2 case labels have same value.

4. 2 case labels may be associated with same statement.

5. default label is not required. If value of exp. does not match with any label, control transfers outside of switch statement.

6. There can be Almost one default label. It may be coded anywhere.

(2) else if statement
When selection is based on range of values, cond. is not an int.
It is only used when:
(i) selection variable is not integral
(ii) some variable is being tested in exp.

Page No. TAYAL
Date

1. It is a compile error if types in prototype declaration & fn definition are incompatible.

Ex= double divide (int dividend, int divisor);
---

double divide (float dividend, float divisor)
{

}

2. It is compile error to have a diff no. of actual parameters in fn call than there are in prototype declaration.

3. It is a logic error if you code parameters in wrong order.

Ex= double divide (float dividend, float divisor);
---

double divide (float divisor, float dividend)
{

}

4. It is compile error to define local variables with same identifiers as formal parameters.

double divide (float dividend, float divisor)
{ float dividend;
}

5. Using a void return with a fn that expects return value or using a return value with a fn that expects a void return is a compile error.

Page No. TAYAL
Date

6. Forgetting semicolon at end of fn prototype declaration is a compile error. Similarly, using a semicolon at end of header in fn def. is a compile error.

7. It is a run time error to code a fn call without parenthesis, even when it has no parameters.

printHello ; ✗

printHello( ); ✓

8. Logical operator require 2 ampersands (&&) or 2 bars (||). It is logical error to code them with only one.

☆ Loops	Pretest Executes	Posttest Executes
Initialization	1	1
no. of tests	n+1	n
action executes	n	n
updating executes	n	n
minimum iterations	0	1

n is no of iterations

While & for loops are pretest loops & do...while loop is posttest loop.

For loop is usually used for counter controlled loops while & do...while are used for event controlled loops.

Page No.
Date

1. While Loop

```
while (exp)
    statement
```

No semicolon is req. at end of while statement. If you see a semicolon at end of code, it actually belongs to statement within while statement.

There are 3 cond. that will cause extraction to fail: end of data known as end of file is reached, Invalid numeric data is entered or a read error occurs.

2. For loop

A for loop is used when your loop is to be executed a known no. of times.

```
for (exp1 ; exp2 ; exp3)
    statements
```

```
i=1;
sum=0;
while (i<=20)
{ cin>>a;
  sum+=a;
  i++;
}
```

```
sum=0;
for(int i=1; i<=20; i++)
{ cin>>a;
  sum+=a;
}
```

Page No. TAYAL
Date

3. Do... while loop

```
do
{ statement 1
  statement 2
} while (exp);
```

Because do... while limit test is not done until end of loop, we use it when we know that body of loop must be done at least once.

* Whenever a variable is used only within a block, it should be defined in that block.

☆ Jump statements
1. break
2. Continue
3. Return
4. Goto

1. Break Statements
In a loop, break statements causes a loop to terminate.

```
while (cond)
{ for (...;...;...)
  { if (other cond.)
      break;
  } ...
}
```

/* Break statement takes you out of inner loop (the for loop). The while loop is still active */

Page No. ____ TAYAL
Date | | |

2. Continue statement

The continue statement is used to skip rest of statements in a loop & start a new iteration without terminating the loop.

```
while (limit test)          do                      for(initialization; test; update)
{ ---                       { ---                   { ---
  ---                         ---                     ---
  continue; ---->             continue; ---->         continue; ---->
  ---                         ---                     ---
} //while                   } while (limit test);   } //for
```

★ Recursion

Recursion is a repetitive process in which a fn calls itself.

$$factorial(n) = \begin{cases} 1 & \text{if } n=0 \\ n*(n-1)*\_\_*3*2*1 & \text{if } n>0 \end{cases}$$

Iterative fn def

$$factorial(n) \begin{cases} 1 & \text{if } n=0 \\ n * factorial(n-1) & \text{if } n>0 \end{cases}$$

Recursive fn def

The statement that solves the problem is known as base case.
Every recursive fn must have a base case.
Rest of fn called general case.

Page No. _____
Date

* Every recursive call must either solve part of problem or reduce size of problem.

* Rules for designing a recursive fn
1. First determine base case.
2. Then determine general case.
3. Combine base case & general case into a fn.

☞ The base case, when reached, must terminate without a call to recursive fn, that is, it must execute a return

* It is compile error to omit semicolon after exp. in do... while statement.

* It is a logic error to place a semicolon after exp. in a while or for statement

* It is compile error to use a for statement with commas rather than semicolon,

```
for ( int i=0, i<10, i++)
```

* It is logic error to update terminating variable in both for statement & in body of loop.

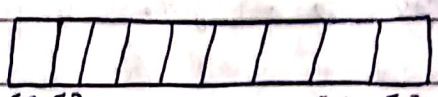
```
for (int i=0; i<10; i++)
{ - - -
  - - -
  i+=1;
}
```

Page No.
Date

☆ Arrays

```
int scores[9];
```

(type of each element)

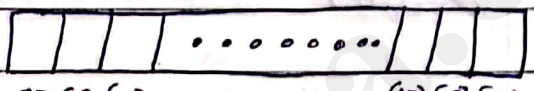



```
char name[10];
```

(name of array)


```
float gpa[20];
```

(no. of elements)



Declaring & defining arrays

address = array address + (sizeof(element) * index)

* When array is going to be completely filled, the most appropriate loop is for loop because no. of elements is fixed & known.

```
for (i=0; i<10; i++)
    cin >> scores[i];
```

otherwise use while or do-while loop.

* In C++, name of an array is a primary exp. whose value is address of first element in array.

Page No. TAYAL
Date

* There are 2 rules associated with passing whole array to a fn.

1. The fn must be called by passing only name of the array.
2. In fn definition, formal parameter must be an array type, size of array does not need to be specified.

```
double average (const int x[]);
int main()
{ double ave;
  int base[5] = { 3, 7, 2, 4, 5 };
    ave = average (base);
    return 0;
}
    double average ( const int x[])
    {   double sum=0;
        for (int i=0; i<5; i++)
            sum += x[i];
        return (sum/5.0);
    }
```

* Sorting

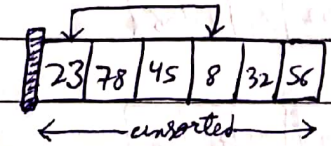
1. Selection Sort

In a selection sort, list is divided into 2 sublists, sorted & unsorted.

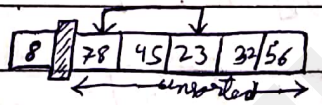
Page No. TAYAL
Date

We find the smallest element from unsorted sublist & swap it with element at beginning of unsorted lists. After each selection & swapping, imaginary wall b/w 2 sublists move one element ahead. increasing the no. of sorted elements & decreasing no. of unsorted ones.

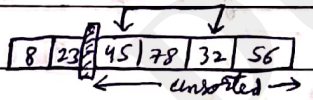
A list of n elements requires n-1 passes to completely rearrange the data.



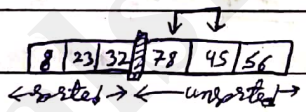
original list



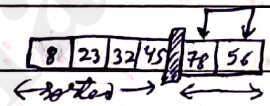
After pass 1



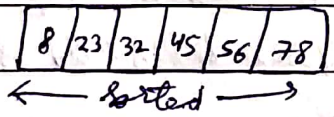
After pass 2



After pass 3



After pass 4



After pass 5

Selection Sort

2. Bubble Sort

In bubble sort method, list is divided into 2 sublist: sorted & unsorted. The smallest element is bubbled from unsorted sublist & moved to sorted sublist. After smallest element has been moved to sorted list, wall moves 1 element ahead, increasing no. of sorted elements &

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

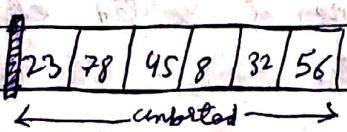
- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

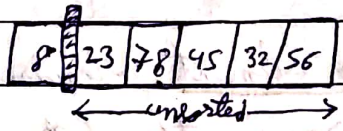
Page No. TAYAL
Date

decreasing no of unsorted ones.

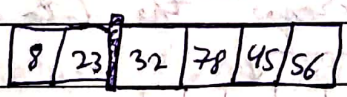
Given a list of n elements, bubble sort req. upto n-1 passes to sort data.



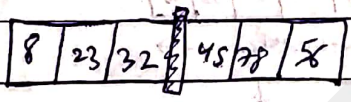
original list



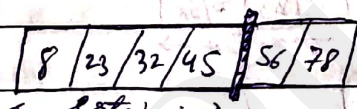
After pass 1



After pass 2



After pass 3



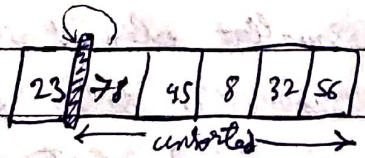
After pass 4

Bubble sort ex.

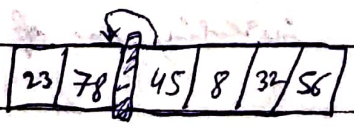
3. Insertion Sort

List is divided into 2 parts: sorted & unsorted. In each pass, first element of unsorted sublist is picked up, transfered to sorted sublist & inserted at appropriate place.

A list of n elements will take at most n-1 passes to sort data.

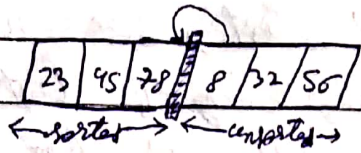


Original list

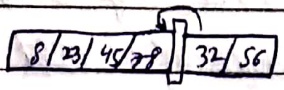


after pass 1

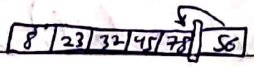
Page No. ——— TAYAL
Date



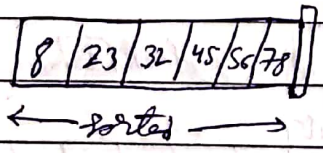
after pass 2



after pass 3



after pass 4



after pass 5

Insertion sort ex.

☆ Searching

1. Sequential Search

The sequential search is used if list being searched is not ordered.
In sequential search, we start searching for target from beginning of list & we cont. until either we find target or we are sure that it is not in list.

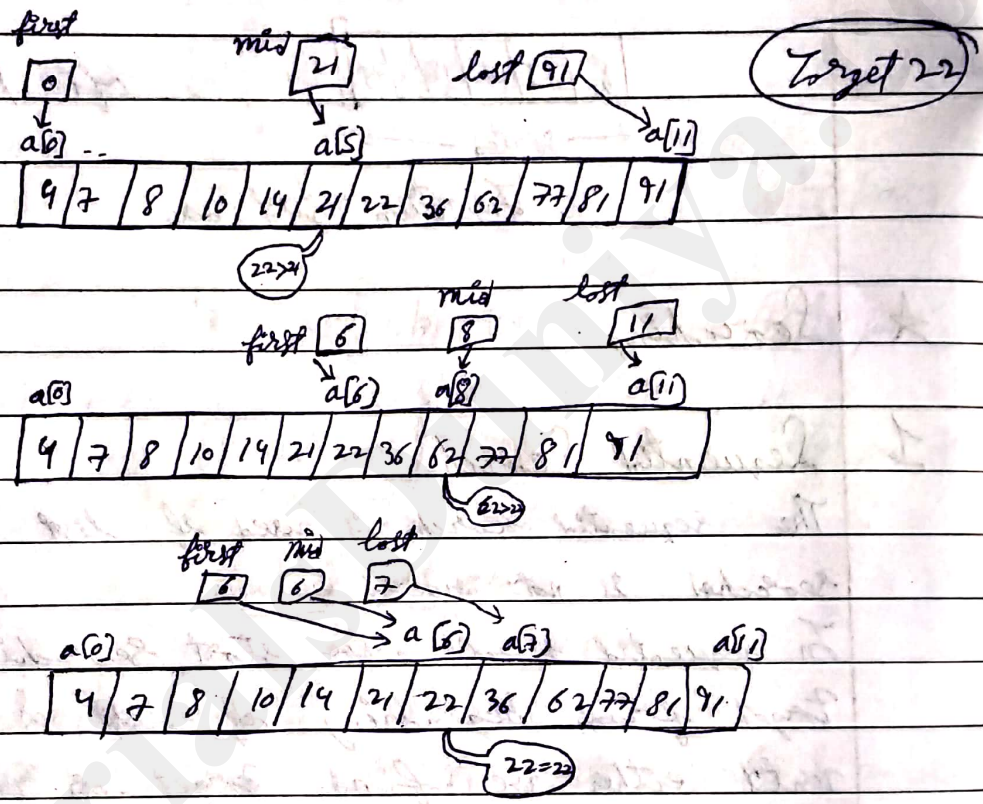
2. Binary Search

Binary search starts by testing data in element at middle of array. This determines if target is in first half or second half of list. If it is in first half, there is no need to check second half & vice versa.

Page No. ____ TAYAL
Date

To find middle of list, we need 3 variables
① one to identify beginning of list.
② one to identify middle of list.
③ one to identify end of list.

$$Mid = (first + last) / 2$$



first 8 mid 6 last 7
for terminates

Binary Search ex.

* 3 things are needed to declare & define an array: its Name, Type & Size.

* To initialize all elements in an array to zero all you have to do is initialize ferst element to zero.

Page No.
Date

* It is compile error to leave out index operator in in statement.

```
float costAy[20];
cin>> costAy          // Invalid
cin>> costAy[0];      // correct code
```

* It is compile error to omit array size in prometer declaration for any array dimension other than first.

* Pointers

Pointer const., drawn from set of address for a computer, exist by themselves.
We can't change them, we can only use them.

The address operator (&) provides a pointer const. to any named location in memory.
Eg &addr

When the ampersand (&) is used as a prefix to a variable name, it means address of variable.
When it is used as a suffix to a type, it means reference prometer.

The address of a variable is address of first byte occupied by that variable.

Page No. ____
Date | | | |

* C++ provides a special null const., NULL, that can be used to set a pointer so that it points to nothing.
* The value of NULL const. is 0.
* Use either 0 or NULL to initialize a pointer does not contain an address.

☆ Indirection operator (*)

*p ;

For ex: We want to add 1 to variable, a.

p = &a

a++, a=a+1, *p = *p+1, ++(*p), (*p)++

* The exp. x, *p, *q all are exp that allow variable to be either inspected or changed.

Before	statement	after
p → q → [3] x	x=4;	p → q → [4] x
p → q → [4] x	x=x+3;	p → q → [7] x
p → q → [7] x	*p=8	p → q → [8] x
p → q → [8] x	*&x = *q + *p	p → q → [16] x
p → q → [16] x	x = *p * *q	p → q → [256] x

Accessing variables through pointers

Page No. TAYAL
Date

The indirection and address operators are inverse of each other, when they are combined in an exp., such as *&x, they cancel each other.

*&a == a

* Pointer declaration & definition

p is a pointer to char, q is pointer to int, r is a pointer to floating point variable.

type* identifier

char* p, int* q, float* r;

int x;
int* p = &x;

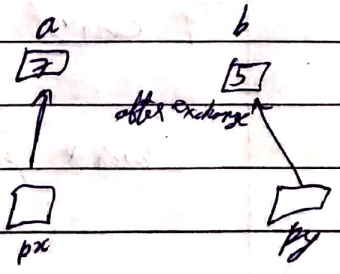
step 1: declare variable → int* p; declaration
step 2: Initialize variable → p = &x; initialization

Define a pointer with an initial null value.

int* p = 0;

The diff. b/w pass by reference & passing pointers is that with pointers an alias is not created — we must use the ~~def~~ dereference operator to effect the change.

```
int a = 5, b = 7;
exchange (&a, &b);
```



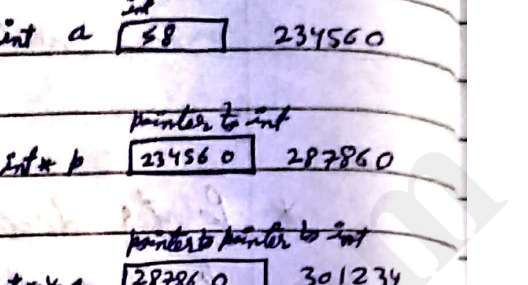

```
void exchange (int* px, int* py)
{ int temp = *px;
  *px = *py;
  *py = temp;
  return; }
```

Page No.
Date

☆ Pointers to pointers

```
int a; int* p; int** q;
a = 58;
p = &a;
q = &p;
cout << a << " ";
cout << *p << " ";
cout << **q << " ";
```

58
58
58



✱ Compatibility

It is invalid to assign a pointer of one type to a pointer of another type.

In C++, we can't use assignment operator with pointers to diff types; if we try to, we get a compile error.

The exception to this rule is void pointer.

The void pointer, known as universal or generic pointer, can be used with any pointer, and any pointer can be assigned to a void pointer.

✱ Since a void pointer has no object type, It can't be ~~[illegible]~~ dereferenced.

```
void* pvoid;
```

Page No. _____
Date | | | |

* Casting pointers

To cast a char pointer, p, to point to an integer (a), you could cast it as shown below.

```
int a;
char* p;
p = static_cast<char*>(&a);
```

```
void* pvoid;   char* pchar;   int* pint;
pvoid = pchar;
pint = pvoid;
pint = static_cast<int*>(pchar);
```

```
char c='z';        char* pc;
int a=58;          int* pa;        int** ppa;
pc = &c;           // valid
pa = &a;           // valid
ppa = &pa;         // valid

pc = &a            // Invalid
ppa = &a;          // Invalid
```

* Address of memory location is a pointer const & can't be changed.

* You must not use a pointer variable before it has been assigned the address of a variable.

```
int* p;
x = *p;            // Error
*p = x;            // Error
```

Page No.
Date

* A pointer variable can't be used to refer to a nonexistent pointed variable.

```
int *p;
cin >> *p;        // Error because *p does not exist
```

* Do not dereference a pointer variable of type void *.

* The address operator (&) can only be used with an lvalue. The result is rvalue.

* In the following definition, only first variable is a pointer to an integer, Rest are integers.

```
int *ptr, ptr1, ptr2;
```

* It is compile error to initialize a pointer to numeric const.

* It is compile error to assign an address to any variable other than a pointer.

```
int *ptr = 59;
int x = &y;        // Error
```

* It is a logic error to dereference a pointer whose value is NULL.

* If ary is an array, then
  ary is same as &ary[0]

* If ary is name of an array, then
  ary[i] is same as *(ary+i)

Page No. TAYAL
Date

* It is an error to use pointer arithmetic with a pointer that does not reference an array.

* It is a logic error to use subtraction on 2 pointers that are referencing diff arrays.

* It is a compile error to subtract a pointer from an index.

* It is a compile error to attempt to modify name of an array using pointer arithmetic.

```
table ++ ;  //Error: table is const.
```

* It is compile error to use pointer arithmetic with multiply, divide or module operator.

* A pointer variable is a variable that can store an address.

☆ Classes

```
class Fraction
{ private: int numerator, denominator;
  public:
      void store (int number, denom);
      void print () const;
```

Page No.
Date

* Scope
class_name :: member_name

Scope resolution operator (::)
It is used when an identifier that we refer to is not within current scope.

* Class objects

Fraction fr;

General format for member access is object identifier, dot, member identifier.

objectID.memberID

* Use scope resolution operator to refer to a specific class type & use member operator to refer to data or fn within an instantiated class object.

* This pointer
When we call a class member fn, it automatically contains a hidden pointer known as this.
The This pointer contains the address of invoking object so that it can refer to data & other fn in class.

Page No. TAYAL
Date

* Characteristics of this pointer
1. It is a const. pointer, which means that we cannot change its contents.
2. Most of the time it is used implicitly - that is, it is hidden.
3. We can use it explicitly when we need to access current host object.

*this.numerator
*this.store(2,10)

* Function in class can be divided into 3 categories:-

① Manager functions
It creates, copy or destroy objects.
The manager fn are constructor, copy constructor, & destructor.

② Mutator fn
It can change the state of invoking object.

③ Accessor fn
It cannot change state of invoking object, it is normally defined as a const fn

Accessor & mutators must always have a return type.

* Classes should be invariants : 2 instances of class with same value should have same representation.

Page No. ____ TAYAL
Date

★ Constructor

Constructors are special member functions that are called when an instance of a class is created.

★ Uses of constructor

1. A constructor initializes class data members when required.
2. Constructors can be used to validate data of the class & thus make program more robust.
3. A constructor must be used to allocate memory for an object.

The name of a constructor is same as name of class & it may not have a return type.

Format

```
classname (parameter list);   // declaration

classname :: classname (parameter list)   // definition
{
   ...
}
```

★ Overloaded constructors

When we need multiple constructors, we implement them as overloaded constructors.

Page No. ____ TAYAL
Date

(1) First, default constructor, is used whenever an object is instantiated with a value of zero.

```
Fraction :: Fration()
{  numerator = 0;
   denominator = 0;
}
```

(2) Second constructor allows us to create a fraction with an integral value. It requ only one parameter.

```
Fraction :: Fraction (int numer)
{  numerator = numer;
   denominator = 1;
}
```

(3) Third initializes fraction by setting numerator & denominator to values passed as parameters.

```
Fraction :: Fraction (int numer, int denum)
{  numerator = numer;
   denominator = denum;
```

A class must have at least one constructor, either defined by program or by compiler.

★ Default constructor

Default constructor is called whenever an object is created without passing any arguments.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

Page No.
Date

① If a programmer doesn't provide any constructor compiler provides a default constructor, ~~which~~

② If we provide any type of constructor, then compiler doesn't provide a default constructor.

* There is only one default constructor, it connot be overloaded.

★ Copy Constructor
A copy constructor is a function that is called whenever a copy of an existing instance needs to be created.

```
Fraction (const Fraction& fr);   // declaration
Fraction :: Fraction (const Fraction& fr) // definition
{   numerator = fr.numerator;
    denominator = fr.denominator;
}
```

We pass a const of existing object to fn by reference. We pass the existing object by reference, not by value, to avoid creating an extra copy.
We pass the existing object as a const because we do not want fn to be able to change state of object being passed - that is to be able to change value of its data members

Page No. ——— TAYAL
Date | | | |

* Bitwise copy
Object's contents are simply copies bit by bit from original object to new object.

* Logical copy
It makes a true copy of structure, including its dynamic structure if necessary.

* default copy constructor
If we do not define a copy constructor, system provides a bitwise one.

* There is always one copy constructor avilable in a class.

A copy constructor is used:-
(1) When an object is instantiated using another object as a parameter.
(2) When an object is passed by value to a fn
(3) When an object is returned from a fn.

Instantiated object :- When a class object is explicitly passed as a parameter during instantiation.

Passed objects:- When an object is passed to a fn by value, a local copy must be created. To create local objects, copy constructor is called.

Page No. _____
Date | | | |

Returned objects :- When an object is returned from a fn, it must be copied to a receiving object or inserted in an exp.

* Always pass objects to a fn by reference, when object cannot be changed, pass it as a const.

* Always return object by value.

* Destructor
Name of destructor is name of class preceded by a tilde (~).
Like a constructor, a destructor cannot have a return type.
However, unlike a constructor, argument list of a destructor must be empty. This means we cannot overload a destructor.

A class can have one and only one destructor.

```
Fraction :: ~Fraction()
{
}
```

* Default destructor
If we do not provide a destructor for our class system provides one for us.
It does nothing.

Page No. _____ TAYAL
Date

① Unary class fn
A unary class fn uses only one instance of class.
It does not have any parameter of class type.
fn1.print();

② Binary class fn
A binary class fn uses 2 instances of class.
If only one of instance is passed to fn, func. can be implemented as member fn.
If both of instances are passed as parameter.
Then fn is better implemented as friend fn.

* Friend fn
If a fn is a friend of a class, it can have access to all members of class, even the private ones.
* A fn can only be declared a friend by class itself.

* Friend fn are not members of a class, but are associated with it.

Prototype declaration
class fraction
{ ....
....
public: friend fraction add(const fraction f1,
const fraction f2);
}

Page No.
Date

* If is a compilation error to omit semicolon at end of a class definition.

* If is a compilation error to initialize data variables when defined in a class declaration.

* It is a compile error to access a private member outside the class.

* Class declaration allocates no memory space for variables — they must be defined & initialized by constructors.

* We can have more than one constructor, but only one copy constructor & only one destructor.

* Inline Functions

In an inline fn, rather than transfering control to a called fn, the called fn's code replaces call in calling fn.
This substitution of replicated code in place of call is made by compiler.

Inline fn are used to improve efficiency of a program.

An inline recommendation can be made explicitly or implicitly.

Page No. ____ TAYAL
Date

* Non member fn are always defined explicitly.

Non member inline fn
An explicit recommendation is made by simply adding the fn specifier inline to its prototype declaration.

```
inline void dsit (int num);
void dsit (int num)
{ ---
  -- }
```

Explicit Inline fn

```
class Eroction
{ int numerator, denominator;
    public: inline void print() const;
};
void Efraction :: print() const
{ cout << numerator << "/" << denominator ;
  return; }
```

Implicit inline fn

```
class Eroction
{ int numerator, denominator;
    public: ---
        void print() const
        { cout << numerator << "/" << denominator;
          return; } ;
```

Page No. TAYAL
Date

Initialization list

Fraction :: Fraction (int n, int d)
{ numerator = n;
denominator = d;
}
Assignment operator
	Fraction :: Fraction (int n, int d)
	: numerator (n),
	denominator (d)
	{ ~~ ~~ }
	Initialization list

* There are 2 parts of an initialization list
1. A colon that seprates fn header from initialization ~~list~~ values.
2. A set of variables followed by their corresponding initialization values enclosed in parenthesis.
Member names are seperated from each other with commas

* Overloading
Overloading is definition of 2 or more fn or operators within some scope using some identifier.

```
void increment ();
void addTo (const Fraction& fr2);
friend Fraction add (const Fraction& fr2, const Fraction& fr3);
```

operation	fn syntax	Arithmetic syntax
Increment	fr1.increment()	++fr1;
add to	fr1.addTo(fr2)	fr1 += fr2
add	add (fr2, fr3)	fr2 + fr3

Page No. ——— TAYAL
Date | | | |

Overloading member operators

To convert our fn to overloaded operator, all we must do is change fn name to keyword 'operator' followed by operator we want to use.

With exception of prefix & postfix ++ increment, nothing else in fn is changed.

The prefix & postfix increment require that we also return fraction object.

Prefix Increment & decrement operators

Prototype: Fraction& operator ++();

```
Fraction& Fraction :: operator ++()
{ numerator += denominator;
  return (*this);
}
```

Postfix Increment & decrement operators

Prototype: const Fraction operator ++(int)

If overloaded operator definition doesn't have a parameter list, then fn is using increment or decrement, which means that it is prefix operator. If an integer parameter is specified in parameter list, then fn is postfix operator, which means that object is incremented after value is determined.

Page No.
Date

```
const Fraction Fraction :: operator ++(int)
{ const Fraction temp(*this);
  numerator += denominator;
  return temp;
}
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well