**C**

1. **Course Plan**
2. introduction to programming
3. input-output
4. operator
5. math
6. conditional statement
7. switch
8. loop
9. series
10. pattern/pyramic
11. array-one dimensional
12. array-two dimensional
13. string
14. function
15. recursion
16. file
17. structure
18. pointer
19. advance c
20. **How to read C programming**
21. translator programm
22. token
23. input-output
24. operator
25. control statement
26. array
27. function
28. **Translator Programm**
29. source code is written by higher programming language.
30. Machine code or object code is written by translator.
31. Transaltor porgramm convert source code to machine code or object code.
32. Translator programm: compiler, interpreter, assembler
33. assembler convert assembly code to machine code.
34. Compiling language: c,c++,object-c, c#, pascal, cobol,ada, visual basic,smalltalk,scheme
35. interpreting language: basic,php,python,perl,ruby, javascript
36. language processing system: source code→preprocessor→ compiler→ assembler→ linker→ binary executable
37. compiler processing system: source code→ compliling→ object code/list of errors→ input→ object code→ output.exe
38. compiler total program translate at a time then execute.
39. Interpreter line by line translate then execute.
40. Interpreter every statement make a exe file.
41. Comiler vs interpreter:

|  |  |
| --- | --- |
| **compiler** | **interpreter** |
| Whole program translate at a time. | Line by line translate. |
| Whole program error showing at a time. | Line by line showing error. |
| Compiler fast translate. | Interpret slowly translate. |
| Compiler once time compiling finished next no required translate the programm. | Every running need to translate. |
| For big computer used this. | For micro computer used this. |

1. **Algorithm**
2. algorithm means any problem solve step by step.
3. Algorithm must be easyier.
4. Every steps must be clear so that any programmer can understand.
5. Solv problem step by step.
6. Various same type of problem use this.
7. Advantages:
8. easily help to understand this programm objectives.
9. Help us to measuring errors.
10. Programm make flexibility
11. easily written any hard programm.
12. **Flowchart,pseudocode**
13. using flowchart programm can demonstrate with picture or sign.
14. Flowchart 2 types: system flowchart,programm flowchar.
15. Programm flowchart use for compose programm.
16. Algorithm vs flowchart:

|  |  |
| --- | --- |
| **algorithm** | **flowchart** |
| Using algorithm problem solve step by step. | Problem solve stages are demonstrate by diagram. |
| Algorithm describe based. | Flowchart diagram based. |
| For understanding programm need to much time. | For understanding programm need ot short time. |
| Programm flow of porcess hard to understand. | Programm flow of process easy to undersand. |
| Remove porgramm error is to hard | Remove prograam error is easy. |

1. Pseudocode word means same format of code.
2. Before writing programm any lnaguage programmer make a text based code this is called pesudocode.
3. **Stages of programm making and debuging**
4. problem selecting→ problem analysis→ programm desing→ programm development→ programm implementation→ programm maintaining
5. bug=programm errors called bug.
6. Debug= programm errors solved called debug.
7. Programm errors: syntax error, logical error information error.
8. **Introduction C Programm**
9. c invended Dennis Ritchie 1972
10. form c origin – c++, c#, java, perl, php, javascript
11. characterastics:
12. general purpose language.
13. Middle level language.
14. System programming language.
15. Structural language.
16. Platform dependent.
17. Case sensitive.
18. C use case: os,translator,editor,games,virus,anti-virus
19. **C Programm Environment**
20. editor,compilar,debugger
21. ide
22. **First C Programm**
23. header file
24. main function
25. include
26. stdio=standard input and output
27. return data type value;
28. getchar() = clean output
29. **C Comments and Escape Sequence**
30. comments: // or /\*\*/ or single line or multi line comments
31. escape sequence: \n, \t, \\
32. **C Keywords,Variables,Data Types**
33. In c programming almost 32 keywords have as a reserved word.
34. **Keywords:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Auto** | **Double** | **Int** | **struct** |
| **Break** | **Else** | **Long** | **switch** |
| **Case** | **Enum** | **Register** | **typedef** |
| **Char** | **Extern** | **Return** | **union** |
| **Const** | **Float** | **Short** | **unsigned** |
| **Continue** | **For** | **Signed** | **void** |
| **Default** | **Goto** | **Sizeof** | **volatile** |
| **Do** | **If** | **Static** | **while** |

1. Keyword use lowercase letter.
2. **Data types:**
3. char = 1 bytes
4. int = 4 bytes
5. float = 4 bytes
6. double = 8 bytes
7. memory management:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Type** | **Examples** | **Keywords** | **Storage Size** | **Format Specifier** |
| charecter | ‘a’ | char | 1 byte | %c |
| number | 125 | int | 4 bytes | %d |
| floating | 6 deimal places | float | 4 bytes | %f |
| double | 15 decimal places | double | 8 bytes | %lf |

1. **Variable:**
2. variable declaration & initialization
3. identifiers:
4. identifiers use A-Z,0-9,a-z,$,\_
5. indentifiers can’t start with number.
6. Reserved word can’t use a indentifiers.
7. Into identifiers can’t have empty space.
8. Indentifiers maximum use 32 characters but recommended 8 characters.
9. **C Input & Output**
10. input: scanf();
11. output: printf();
12. **C sizeof()**
13. sizeof()=getting memory size of this data type
14. **C ASCII Value**

|  |  |  |
| --- | --- | --- |
| **a-z** | **97-122** | **dif-32** |
| **A-Z** | **65-90** |
| **0-9** | **48-57** |  |

1. **Module-03: Input - Output**
2. **Get user input using scanf**
3. scan()
4. scanf after space not accepted string input
5. **Get user string input using fgets,getchar**
6. fgets(name,sizeof,stdin)
7. name = getchar();
8. fgets all character accepted
9. **get data size using sizeof()**
10. sizeof();
11. **convert ascci into value**
12. %c
13. 0-9 = 48-57;
14. A-Z = 65-90;
15. a-z = 97-122
16. ! = 33
17. # = 35
18. $ = 36
19. & = 38
20. \* = 42
21. ^ = 94
22. ( = 40
23. ) = 41
24. @ = 64
25. **convert value into ascci**
26. %d
27. **convert lower to upper case**
28. char-32
29. **convert upper to lower case**
30. char + 32
31. **convert ascii lower to upper using funciton**
32. #include<ctype.h>
33. toUpper();
34. toLower();
35. **format spcifier list**

|  |  |  |  |
| --- | --- | --- | --- |
| **Hole** | **Description** | **Example** | **Result** |
| %d,%i | Print any integer |  |  |
| %x,%o | Print any integer in hex or octal formate |  |  |
| %s | Print any string |  |  |
| %f | Print any floating point number |  |  |
| %c | Print any character |  |  |
| %b | Print any boolean |  |  |
| %O | Print any object |  |  |
| %A | Print anything |  |  |
| %lld | Print any long long interger |  |  |
| %lf | Print any double |  |  |

1. **Conversion between decimal and octal and hexa-decimal**
2. %d = decimal
3. %o = octal
4. %x = hexa decimal
5. **Module-04: Operator,Operand, Expression**
6. **introduction**
7. operator
8. operand
9. constant
10. expression
11. **kinds of operator**
12. arithmetic operator
13. assignment operator
14. relational operator
15. logical operator
16. conditional operator
17. unary operator
18. bitwise operator
19. special operator
20. **arithmetic operator**
21. addition(+)
22. subtraction(-)
23. mutiplication(\*)
24. division(/)
25. reminder(%)
26. **math formula convert into c**

|  |  |
| --- | --- |
| **Math formula** | **C program** |
| X=a2-2ab-b2 | X=a\*a-2\*a\*b-b\*b |
| Y=AB2+c | Y=A\*B\*B+c |
| X=Y | Y==Y |
| D=x÷y | D=x/y |
| D = √(b2-4ac)÷2a | D=sqrt(b\*b-4\*a\*c)/2\*a |

1. **Prefer of sign working**
2. ()=> /\*%=>+- => =
3. **calcualte average of some numbers**
4. sum\_numbers/count\_numbers = average
5. **type casting two way**
6. (float)7/2
7. 7\*1.0/2
8. **calculate triangle area**
9. (1/2)\*base\*height
10. **two numbers summation algorithm**
11. *start*
12. *take input two numbers*
13. *summation of two numbers*
14. *output*
15. *end*
16. **two numbers summation flowchart**

start

Input

num1,num2

sum = num1+num2

Output

sum

end

1. **two numbers summation c program**

#include <stdio.h>

int main(){

int num1,num2;

scanf("%d %d",&num1,&num2);

int sum = num1,num2;

printf("%d",sum);

return 0;

}

2. **three numbers summation and average algorithm**
3. *start*
4. *take input 3 numbers*
5. *summation of 3 numbers*
6. *average of summation numbers*
7. *output*
8. *end*
9. **three numbers summation and average flowchart**

start

Output sum,avg

end

Take input num1,num2,num3

Summation of 3 numbers

sum = num1+num2+num3

Average of 3 numbers

avg = sum/3

1. **three numbers summation and average c program**

#include <stdio.h>

int main(){

int num1,num2,num3;

scanf("%d %d %d",&num1,&num2,&num3);

int sum = num1+num2+num3;

float avg = sum\*1.00/3;

printf("%d %f",sum,avg);

return 0;

}

2. **Module-5: Math**
3. abs() - absolute value
4. **#include<math.h>**
5. sqrt() - get square root value
6. pow() - get power of any value
7. log()
8. log10()
9. exp()
10. sin()
11. cos()
12. tan()
13. round()
14. trunc()
15. ceil()
16. floor()
17. **Module-6: assignment operator**
18. = assign
19. += addition assign
20. -= subtraction assign
21. \*= multiplicaiton assign
22. /= divission assign
23. %= modulus assign
24. **module-7: unary operator**
25. + uanry plus
26. - unary minus
27. ++ increment
28. -- decrement
29. unary operator almost used for getting absolute value
30. +a,-a also example for unary operator usecases
31. **module-8: relational operator**
32. < less than
33. > greather than
34. <= less than or equal
35. >= greather than or equal
36. == equal
37. != not equal
38. **module-9: control statement flowchart**

start

input

NO

NO

output

condition

condition

YES

YES

output

output

End

1. **how to check a number even or odd value**
2. **how to check a number negetive or positive**
3. **find out maximum number between two numbers**
4. **Module-10: Logical Operator**
5. && = and
6. || = or
7. ! = not
8. **find out maximum number from three numbers**
9. **calculate a leaf year algorithm**
10. start
11. input year
12. is year%400 == 0
13. ` 1) yes,print leaf year
14. 2) no,go to next step
15. is year%4 == 0 && year%100 != 0
16. 1)yes,print leaf year
17. 2)no, print not leaf year
18. end
19. **calculate a leaf year flowchart**

NO

YES

Output

Not leaf year

Output

Leaf year

End

Is

Year%4 == 0

And

Year%100 != 0

Input leaf year

Is year%100 == 0

start

YES

Output leaf year

NO

1. **how to check a letter capital or small?**
2. **How to to check a letter consonant or vowel?**
3. **Calculate pass or fail from getting result**
4. **calculate letter grade from getting result**
5. **Module-11: Switch statement**
6. **uses keyword**
7. switch
8. case
9. break
10. defualt
11. **Module-12: Pointer**
12. **introduction**
13. variable print
14. variable address print
15. & = find out the variable address
16. pointers is a variable that stores/points the address of another variable.
17. **Why pointer?**
18. Pointers are powerful features of C and C++ programming that differentiates it from other popular programming language like: Java and Python.
19. Using pointer makes the software more efficient cause it works with memory management.
20. But excessive usage may make the application less understandable.
21. **Symbol**
22. & symbol is used to get the address of the variable.
23. \* symbol is used to get the value of the variable that the pointer is pointing to.
24. %p= printf of address
25. %p = &a,a = get value of a address or value
26. %d = \*p = derefernce
27. dereference
28. **objective of pointer**
29. using you can change original value