Introduction to programming through C

Why teach C?

- C is *small* (only 32 keywords).
- C is *common* (lots of C code about).
- C is *stable* (the language doesn’t change much).

- C is *quick running*.

- C is the *basis for many other languages* (Java, C++, awk, Perl).
- It may not feel like it but C is one of the easiest languages to learn.

History of ‘C’

- Root of the modern language is

ALGOL 1960. It’s first computer

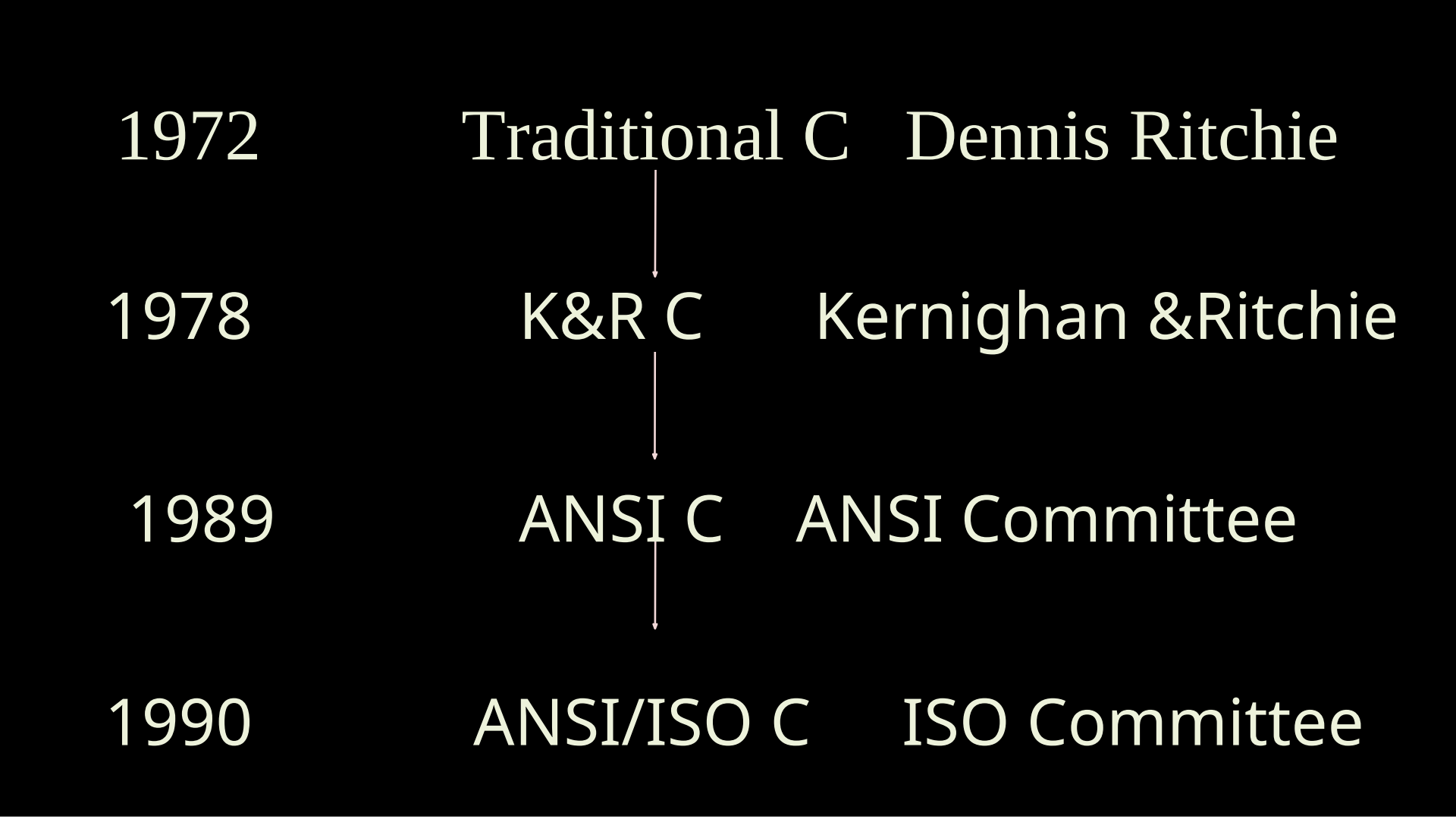
language to use a block structure.

- It gave concept of structured programming.

- In 1967, Martin Richards developed a language, **BCPL** (Basic Combined Programming Language)

In 1970,by Ken Thompson created a language called as ‘B’.
It used to create early version of **Unix.**

In 1972,by Dennis Ritchie introduced new language called as ‘C’ .


1972 Traditional C Dennis Ritchie
1978 K&R C Kernighan &Ritchie
1989 ANSI C ANSI Committee
1990 ANSI/ISO C ISO Committee

Features Of ‘C’

It is a robust language.

Programs written in ‘C’ are efficient and fast.
(Because of variety of data types and powerful operators)

Highly Portable. (related to OS)
Well suited for structured programming.
Ability to extend itself.

Program & Programming Language

Program:- A Set of instructions which

When carried out by processor for some Specific input, generates specific output.

Programming language:- A specific manner of writing a program with some Predefined rules, symbols & their use as

a part of language.

i.e. Pascal, C, C++, VC++, JAVA, VB.

Basic structure of ‘C’

1) Documentation Section :-

   It has set of comment lines(name of program, author details).

   What is Comment line??

- To guide a programmer. To write a note for function,operation,logic in between a program.
- Non-executable statement.

- Can’t be nested.

e.g:- /* Hello /* abc */ Hi */

ERROR.

2) Link Section :-

It provides instructions to the compiler to link

function from the system library.

 # include Directive:-

- To access the functions which are stored in the library, it is necessary to tell the compiler , about the file to be accessed.

Syntax:-

#include<stdio.h>

- stdio.h is header file.

3) Definition Section

It defines all symbolic constants.

#define instruction defines value to a symbolic constant.

#define:-

4) Global Declaration Section

Some variables that are used in more than one function, such variables (global variables) declared in the global declaration section.

It also declares all the user-defined function.

5) Main() function section:

Every ‘C’ program must have one main() function section.

It contains two parts

1) Declaration part:

2)

Hello World in C

```
#include <stdio.h>
void main()
{
  printf(“Hello, world!\n”);
}
```

Preprocessor used to share information among source files

- Clumsy
+ Cheaply implemented
+ Very flexible

Hello World in C

Program mostly a collection of functions
"main" function special: the entry point
"void" qualifier indicates function does not return anything

```
#include <stdio.h>
void main()
{
  printf("Hello, world!\n");
}
```

I/O performed by a library function: not included in the language

LinkSecaon

&finiaon Secoon

Globel &clerañon Section

FuncbooSecfon

```
#include<stdio.hi /’link ’/

#define pi 3.14

/’definition’/ flDat area,

/*g2obai ’/

VDid main()

        flDat r,
        r=l.0;
        area   i’r’r;
```

Main()
{
____DECLERATION

PART_

____

____
____EXECUTABLE
PART_

’’Declaration’’

General structure of a C Program

```
}
```

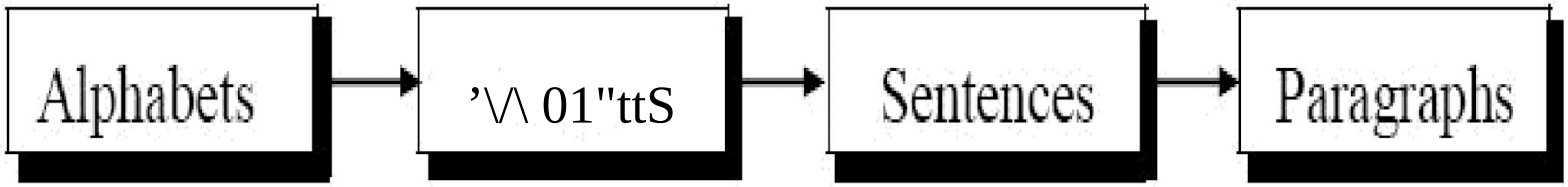
```
printf(”area is %d
\n”,area);
```

DocumentafonSecfon

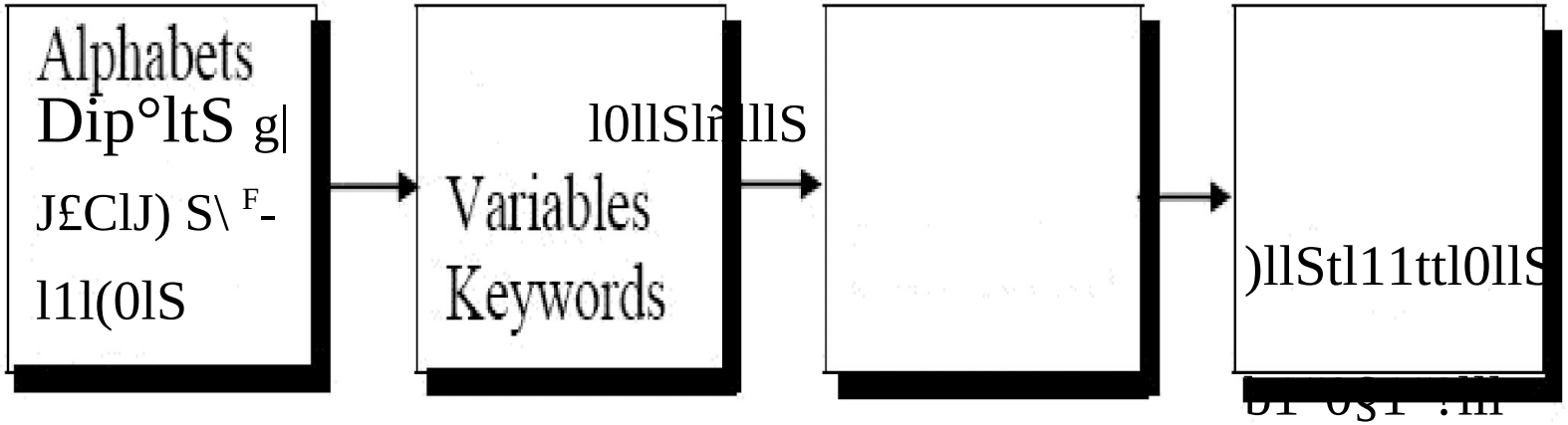
Subprogram Section

Function 1
Function 2

$IC§S 1ll 10?1Ill1l,° (11§11tll lhllgllhgP'



$ttQS 1ll l03lllllllg (:



(lq*PI0 1.1

The C Character Set

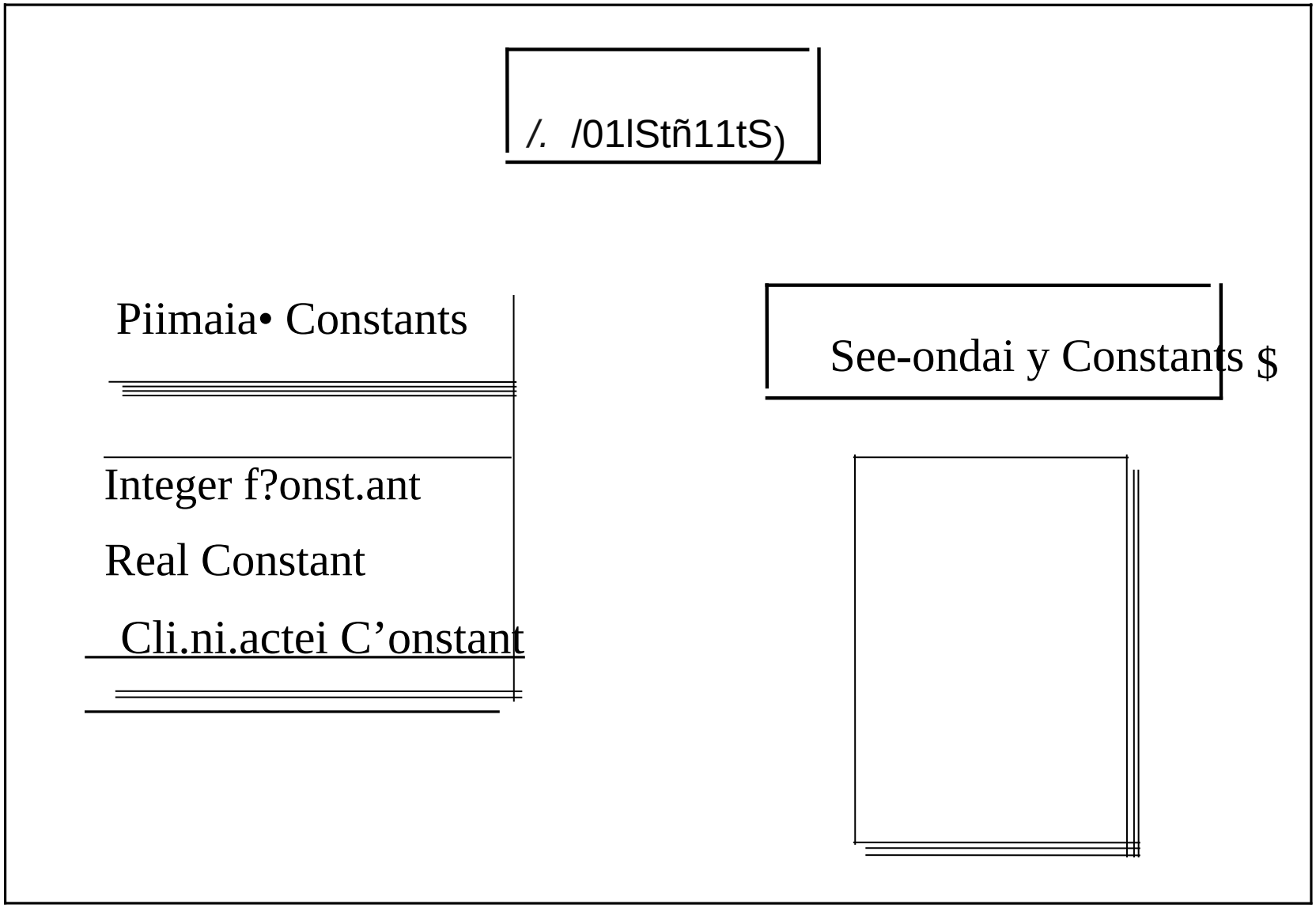
Alphabets	A, B, ….., Y, Z a, b, ……, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /

Figure 1.2

Constants, Variables and Keywords

- The alphabets, numbers and special symbols when properly combined form constants, variables and keywords.

- A constant is an entity that doesn’t change whereas a variable is an entity that may change.


/. /01lStñ11tS)
Piimaia• Constants
See-ondai y Constants $
Integer f?onst.ant
Real Constant
Cli.ni.actei C’onstant

Flsure 1.4

Rules for Constructing Integer Constants

- An integer constant must have at least one digit.
- It must not have a decimal point.
- It can be either positive or negative.

- If no sign precedes an integer constant it is assumed to be positive.
- No commas or blanks are allowed within an integer constant.
- The allowable range for integer constants is

  -32768 to 32767.

Real / Floating Constants

- A real constant must have at least one digit.
- It must have a decimal point.
- It could be either positive or negative.

- Default sign is positive.

- No commas or blanks are allowed within a real constant.
- Ex.: +325.34

426.0

-32.76

-48.5792

constructin re constant expressed in exponential

- The mantissa part and the exponential part should be separated by a letter e.
- The mantissa part may have a positive or negative sign.
- Default sign of mantissa part is positive.
- The exponent must have at least one digit, which must be a positive or negative integer. Default sign is positive.
- Range of real constants expressed in exponential form is

-3.4e38 to 3.4e38.

- Ex.: +3.2e-5
  4.1e8
  -0.2e+3

  -3.2e-5

Rules for Constructing Character Constants

- A character constant is a single alphabet, a single digit or a single special symbol enclosed within single inverted commas.
- Both the inverted commas should point to the left. For example, ’A’ is a valid character constant whereas ‘A’ is not.
- The maximum length of a character constant can be 1 character. Ex.: 'A'

  'I'

  '5'

'='

Rules for Constructing Variable Names

- A variable name is any combination of 1 to 31 alphabets, digits or underscores.
- The first character in the variable name must be an alphabet or underscore.
- No commas or blanks are allowed within a variable name.
- No special symbol other than an underscore (as in gross_sal) can be used in a variable name. **Ex.:**
  si_int
  m_hra
  pop_e_89

C Keywords

Backslash character constants:

'\b'

•yg •

'\n'

'\r'

"*

'\v'

’xt

'\''

'\"'

```
carr′imgard4
m
```

```
p oB
<stdio.h g
                                    float
                                    area;
void main()
```

```
bDV1ZO#tb    t          clocb1e E ,
varticaL  t&            x= l . 0 ;
      §  @
         area=PI’r’r;
```

```
printf("Area is
\n",area);
```

double quote

'\?'	question mark

Output function printf ()

- Following are some examples of usage of

printf( ) function:

- printf ("%f", si) ;

- printf ("%d %d %f %f", p, n, r, si) ;

- printf ("Simple interest = Rs. %f", si) ;

- printf ("Prin = %d \nRate = %f", p, r) ;

- printf ("%d %d %d %d", 3, 3 + 2, c, a

```
+ b * c – d );
```

Format Specifies

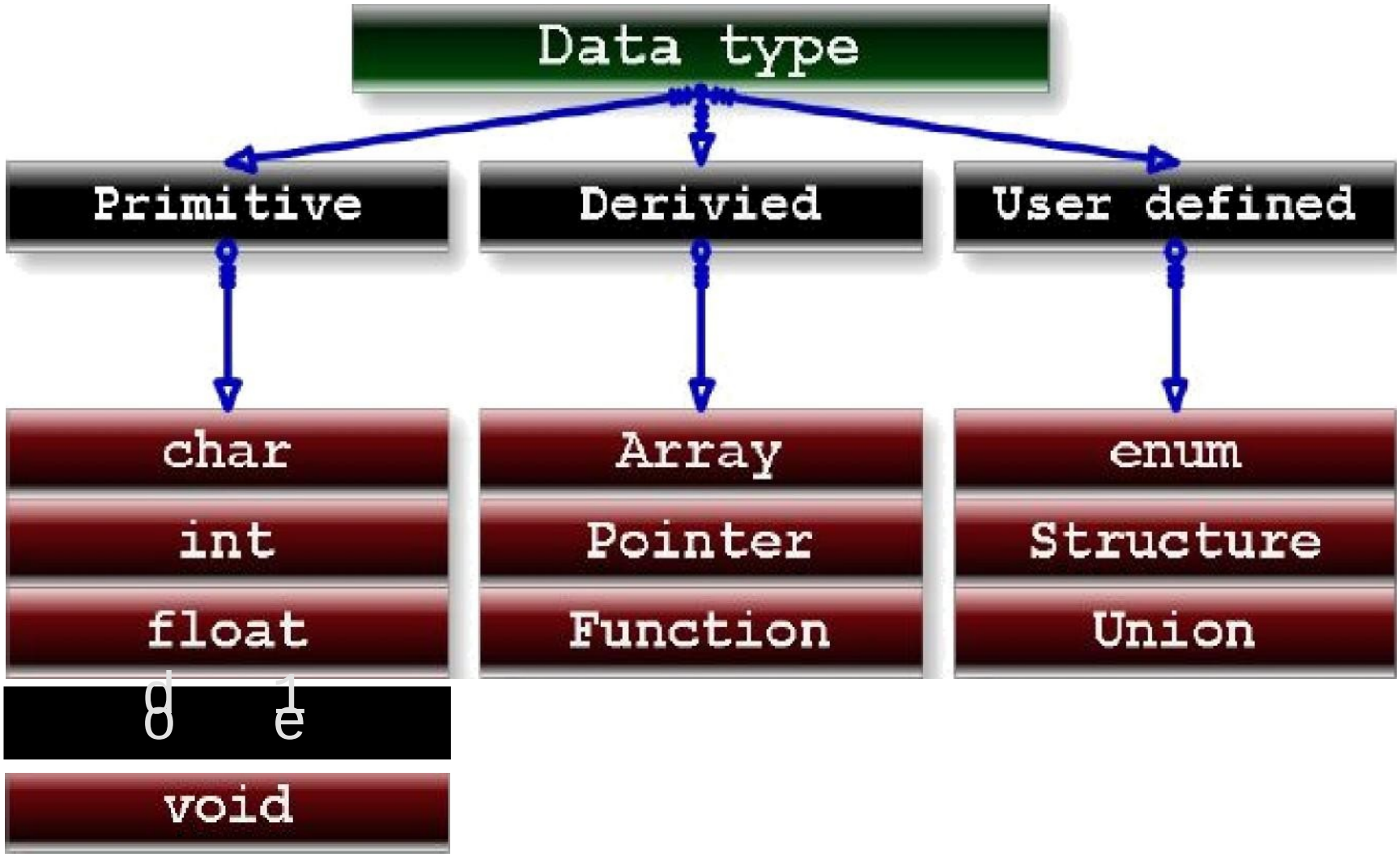
%d %u %6d			
%f %6f %.2f %6.2f	print as floa point	int	printf('Area is #d #n',area); Lteger(ciecimnl, hexa or octa cleciinal) Deciixuil integer. Wilsigyteñ ctecinuil integer.
%o	Octal		
%x			Aeciinal integer, at least 6 characters wicte.
%c or %s			
%%	For printing %		fi P°
	Exponential notation		floating point, at least d characters wicle

Coating point, 2 characters after decimnl point Coating point, at least 6 wiñe ancl 2 altm decimnl

°.4 e, %E

Hexa

cleci
mal
Chara
cter ,•
•
Shnv
g


Data type
Primitive
Derivied
User defined
char
int
float
double
void
Array
Pointer
Function
enum
Structure
Union

Receiving Input : scanf ( ) function

- **printf( ) outputs the values to the screen whereas scanf( ) receives them from the keyboard.**
- For example:

```
printf ( "Enter values of p, n, r" ) ;

scanf ( "%d %d %f", &p, &n, &r ) ;
```

- The ampersand (**&)** before the variables in the scanf( ) function is a must.
- & is an '**Address of' operator**. It gives the location number used by the variable in memory.
- When we say &a, we are telling scanf( ) at which memory location should it store the value supplied by the user from the keyboard.

C Instructions

- Type declaration instruction
  - To declare the type of variables used in a C program. **Ex.: int bas ;**
- Arithmetic instruction

- To perform arithmetic operations between constants and variables.

- Control instruction

- To control the sequence of execution of various statements in a C program.

Type declaration instruction

- The following statements would work int a, b, c, d ;

  a = b = c = 10 ;

- However, the following statement **would not**

work

int a = b = c = d = 10 ;

?????we are trying to use **b (to assign to a) before defining it.**

- Ex.:

```
 int ad ;
float kot, deta, alpha, beta, gamma
;
a
d

=

3
2
0
0

;
```

- Here,

Arithmetic instruction

=

;

0

deta = alpha * beta / gamma + 3.2 * 2 / 5 ;

.

- ***, /, -, + are the arithmetic operators.**
- **= is the assignment operator.**
- 2, 5 and 3200 are integer constants.
- 3.2 and 0.0056 are real constants.
- ad is an integer variable.
- kot, deta, alpha, beta, gamma are real variables.

3types ofArithmetic Instructions

- Integer mode arithmetic statement - Ex.:

```
int i, king, issac, noteit ;

i = i + 1 ;
king = issac * 234 + noteit - 7689 ;
```

- Real mode arithmetic statement - Ex.:

```
float qbee, antink, si, prin, anoy, roi ;

qbee = antink + 23.123 / 4.5 * 0.3442 ;
si = prin * anoy * roi / 100.0 ;
```

- Mixed mode arithmetic statement -Ex.:

```
 float si, prin, anoy, roi, avg ;

int a, b, c, num ;
si = prin * anoy * roi / 100.0 ;
avg = ( a + b + c + num ) / 4 ;
```

NOTE:----------

- C allows only one variable on left-hand side of **=. That is, z = k * l is legal,**

  whereas k * l = z is illegal.

- In addition to the division operator (/) C also provides a modular division (%) operator. Note that the modulus operator (**%) cannot be applied on a float.**
- Also note that on using % the sign of the remainder is always same as the sign of the numerator. Thus –5 % 2 yields –1, whereas, 5 % -2 yields 1.
- An arithmetic instruction is often used for storing character constants in character variables.

```
char a, b, d ;

a = 'F' ;
b = 'G' ;
d = '+' ;
```

- When we do this the ASCII values of the characters are stored in the variables. ASCII values are used to represent any character in memory. The ASCII values of ‘F’ and ‘G’ are 70 and 71
- Arithmetic operations can be performed on **ints, floats and chars.**

```
char x, y ;

int z ;
x = 'a' ;
y = 'b' ;

z = x + y ;
```

Integer and Float Conversions

?????????? Result-------Integer or Float ??????????

```
float a, b, c ;
int s ;
s = a * b * c / 100 + 32 / 4 - 3 * 1.1 ;
```

k is an integer variable

a is a real variable.

??? What Will be the O/P ???

```
int  i = 2, j = 3, k, l ;
float  a, b ;
k = i / j * j ;
l = j / i * i ;
a = i / j * j ;
b = j / i * i ;
printf( "%d %d %f %f", k, l, a, b ) ;
}
```

Hierarchy of Operations

Example 1.2: Determine the hierarchy of operations and evaluate the following expression:

kk = 3 / 2 * 4 + 3 / 8 + 3

Stepwise evaluation of this expression is shown below:

kk = 3 / 2 * 4 + 3 / 8 + 3	
kk = 1 * 4 + 3 / 8 + 3	operation: /
kk = 4 + 3 / 8 + 3	operation: *
kk = 4 + 0 + 3	operation: /
kk = 4 + 3	operation: +
kk = 7	operation: +

(j) Which of the following statement is wrong

(1) mes = 123.56 ;
(2) con = 'T' * 'A' ;
(3) this = 'T' * 20 ;
(4) 3 + a = b ;

(r) The expression x = 4 + 2 % - 8 evaluates to

(1) -6
(2) 6
(3) 4
(4) None of the above

Jggg9/37i£ @X{l79##i0€	@g]jF98#@0
$a \times b - c \times d$	a * b — c * d
$(m + n)(a + b)$	(a + n) * (a+ b)
$3x2 + 2x + 5$	3 * x * x + 2 * x + 5
a+ b+c d+ e ——————— (a + b + c) / {d+ e)	l) — x /
$\left[\frac{2BY}{d+1} - \frac{x}{3(z+y)}\right]$	3 * (z + y)
2 • b * y / { d +	

Simple Conversions

(1) $y = mx^2$ (2) $y = 10^{-4} e^{-\sin(x)^2/5}$ (3) $y = \frac{a+bx}{x\sin(x^2)}$

(4) $y = \sqrt{\sigma\left(\frac{1+x^2}{|1-x|}\right)\sqrt{\log_{10}(1+p^2)}+6-x}$

```
(1)y = m*x*x;

(2)y = 1.0E-4*exp(-pow(sin(x),2)/5.0);

(3)y = (a+b*x)/(x*sin(x*x));

(4)y = sqrt(sigma(1+x*x)/fabs(1-x)*sqrt(log10(1+p*p))+6)-x;
```

or

```
a = sqrt(log10(1+p*p);

b = sigma(1+x*x)/fabs(1-x);

y = sqrt(a*b+6)-x;
```

Control Instructions

(a) Sequence Control Instruction
(b) Selection or Decision Control Instruction
(c) Repetition or Loop Control Instruction
(d) Case Control Instruction

Decision ----The *if Statement*

if (this condition is true)
execute this statement ;

this expression	is true if
x == y	x is equal to y
x != y	x is not equal to y
x < y	x is less than y
x > y	x is greater than y
x <= y	x is less than or equal to y
x >= y	x is greater than or equal to y

The *if-else Statement*

Nested *if-elses*

```
/* A quick demo of nested if-else */
main( )
{
        int   i ;

        printf ( "Enter either 1 or 2 " ) ;
        scanf ( "%d", &i ) ;

        if ( i == 1 )
                printf ( "You would go to heaven !" ) ;
        else
        {
                if ( i == 2 )
                        printf ( "Hell was created with you in mind" ) ;
                else
                        printf ( "How about mother earth !" ) ;
        }
}
```

Forms of *if*

The if statement can take any of the following forms:

(e)

```
if ( condition )
    do this ;
```

```
if ( condition )
    do this
else
    do this
    and this
```

(f)

```
if ( condition )
    do this
else
    do this
    and this
```

```
if ( condition )
    do this
```

Use of Logical Operators

- C allows usage of three logical operators, namely, &&, || and !.

- These are to be read as ‘AND’ ‘OR’ and ‘NOT’ respectively.
- AND’ and ‘OR’ allow two or more conditions to be combined in an **if statement.**

Example 2.4: The marks obtained by a student in 5 different subjects are input through the keyboard. The student gets a division as per the following rules:

Percentage above or equal to 60 - First division
Percentage between 50 and 59 - Second division
Percentage between 40 and 49 - Third division
Percentage less than 40 - Fail

Write a program to calculate the division obtained by the student.

```
/* Method – I */
main( )

    int  m1, m2, m3, m4, m5, per ;

        gi Ef ( "HEIRS IdItS IE fV9
                SEéj9Cb " ) ;

s°*' (–mi                *   -•s- «+•s›.’ s

        if f per >= 60)

              prinh ( "First division ")

        else

              if ( per >= 5D )

                    prinh ( "SecDnd

                    division" )

              else
```

```
g#F# ( "/19r NBItS I1fIV9 Sflbj9Q ' )

p»i•rri.«.mz.r4
+ms|/s
        if ( pm >= 40)

                    prinh ( "Third

                      division" )

              els

              B.

                  printf ( "Fail" )

}
```

```
if(per >= 60)

      @nJ ( "Fint division" ) ,



if (( per >= @ ) && ( per
      < @ ) ) prind
      ("Second division" )

if(( per >=40 j && ( per <
      @ ) ) prind ("Third
      division" ) ,

if (per < 40)

      printl ( "Fail" ) ,
```

The *else if Clause*

Exnmpl 2.fi: \ a progra t calculat th salar a pe th

— e Vnte m D e e)• s r e —

(O))DWH Nble:

IQ

Gender	Yeai•s nf Service	Qnalifccations	SalarJ‘
Mak	>= 10	Post-Graduate	15000
		Graduate	10000
	“° ID	Post-Graduate	10000
	^- ID	Graduate	7000

			I
Feina)e	?•= 10	Post-Graduate	12000
	>= 10	Rduate	9000
	^- ID	Post-Graduate	10000
	< 10	GrRduate	6000 1

```
main(

    int   yos, qual, sal ;
    printf ( "Enter Gender, Years of Service and . &yos

    if ( g == 'm' && yos >= 10 && qual == 1 )

    else if ( g == 'm' && yos >= 10 && qual == 0 )
        sal = 12000 ;

else if ( g == 'f' && yos >= 10 && qual == 0 )
        sal = 9000 ;
```

eJw•et if f p '1" && y•c>-s- -<z 1 O &&

```
sal = 6000 ;
<yc«a l O J
printf ( "\nSalary of Employee = %d", sal ) ;
```

The ! Operator

- Used to reverse an operation. Example:-

```
! ( y < 10 )
```

- This means “**not y less than 10**”.

- In other words, if y is less than 10, the expression will be false, since (y < 10) is true.
- We can express the same condition as

(y >= 10).

Hierarchy of Operators Revisited

Operators	Type
!	Logical NOT
* / %	Arithmetic and modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	Logical AND
||	Logical OR
=	Assignment

What will be the O/P

fJNARY OPERATORS

InCAT tnPnf (++) *and decirmeiit(-- j o;›erazors*

%ie Inc-1"eitt»ilt(Ut2L-1"eaztei‹t) oye›.xtoz utcze«ses (rtPCI"t?aS»s”) fli» x-»1‹i» of the

1'.uialle l*J' 1.

```
int i=el, j;
```

Operation	Example	Equivalent to	
var++ (ostrrieuimñ)	j=(i+ + 10);	j = i+10; i= i+1;	’j is 2’
++var (1€ulm+1ne1)	j= (++i all);	i= i+l; j= i+10;	j is53
Var—— oe%desre em\	j=(i——’ + 10);	j = i+10; i = i—1;	j is52
——var 10ñ€s1’eumñ)	j=(--i + 10);	i= i-1; j= +10;	j is 1

The Conditional Operators

arguments.

- In fact, they form a kind of foreshortened

if-then-else. Their general form is,

expression 1 ? expression 2 : expression 3

- **if** expression 1 is true (that is, if its value is non-zero), **then** the value returned will be expression 2, **otherwise** the value returned will be expression 3

Example:----

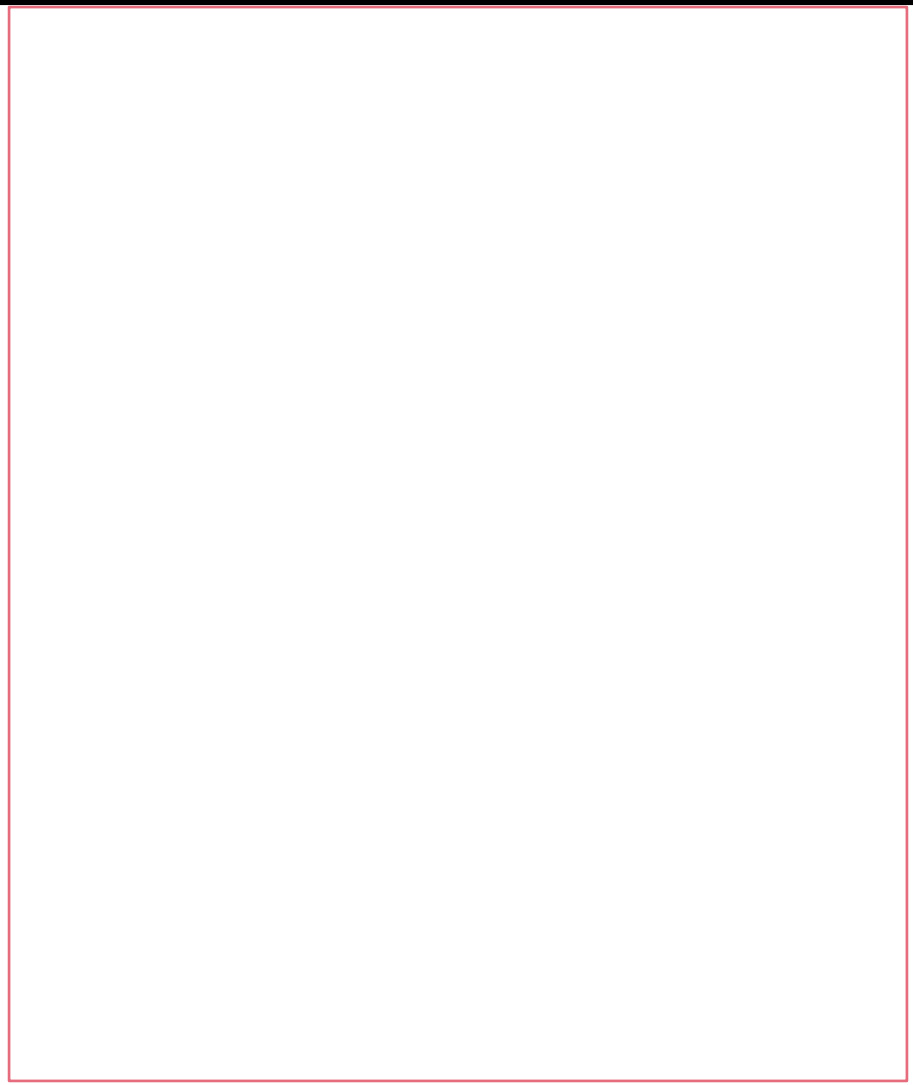
The conditional operators can be nested as shown below:

```
int big, a, b, c;
big = ( a > b ? ( a > c ? 3 : 4 ) : ( b > c ? 6 : 8 ) );
```

????? Whatwill be O/P ??????

Loops

The While loop



— c tes may use t1 or gic
operators as shown in the following examples:

'«vf+ile •. i <= 1 D }

while •: i == 10 && j <= 13

wh ile i j = 1O && i" b < 15 || c <- 2D "i j

while i i == 10 , i
= i -+ 1

_____

while I i <= 1D)

i = i -+ 1

___

_____

int i = 1 :
'while (i < = 1 0

”›

```
        printf ( "%d\n", i ) ;
```

— In.ste.ad nf intcremeltting a lonp counter. xs e can e«-elt decrelttent it and etill tttanage io get the h n‹Jn" Df the loop exec uted repeotedl¿r. This 1s shovslt be1n«'.

```
main( )

    inL  i = 5 ,
    while ( i >= 1
    )

        prinK ( "\nk1ake De cDmputer literate!" j
        i = i - 1
```

— IT is nnt necess.ary- flint U DOE N Otlf1ter ml2S•t only be an intt. It c.un ex'R1t be a Hn.ui.

```
main{ )

    flaaf  a = 1O.0

    while ( a <= 10.5 )
```

```
printf ( "'.nRaindrDps an roses..." ) ;
pñnP ( "...and whiskefs Dn kiLens" ) ,
a = a + O.1
```

7?›'ltat ñO J-"Dt1 think

pcog:rain?

vx ould be tltH D12t@Xtt Of fo llois ing

the

```
main(
    int   i = 1 ,
    while ( i <= 52767 )
    L
        prinN ( "Yoft1n", i )
        i = i • 1
```

— I7+'1›a7 -:ilL be t1›e auty«r a:£ tkc program?

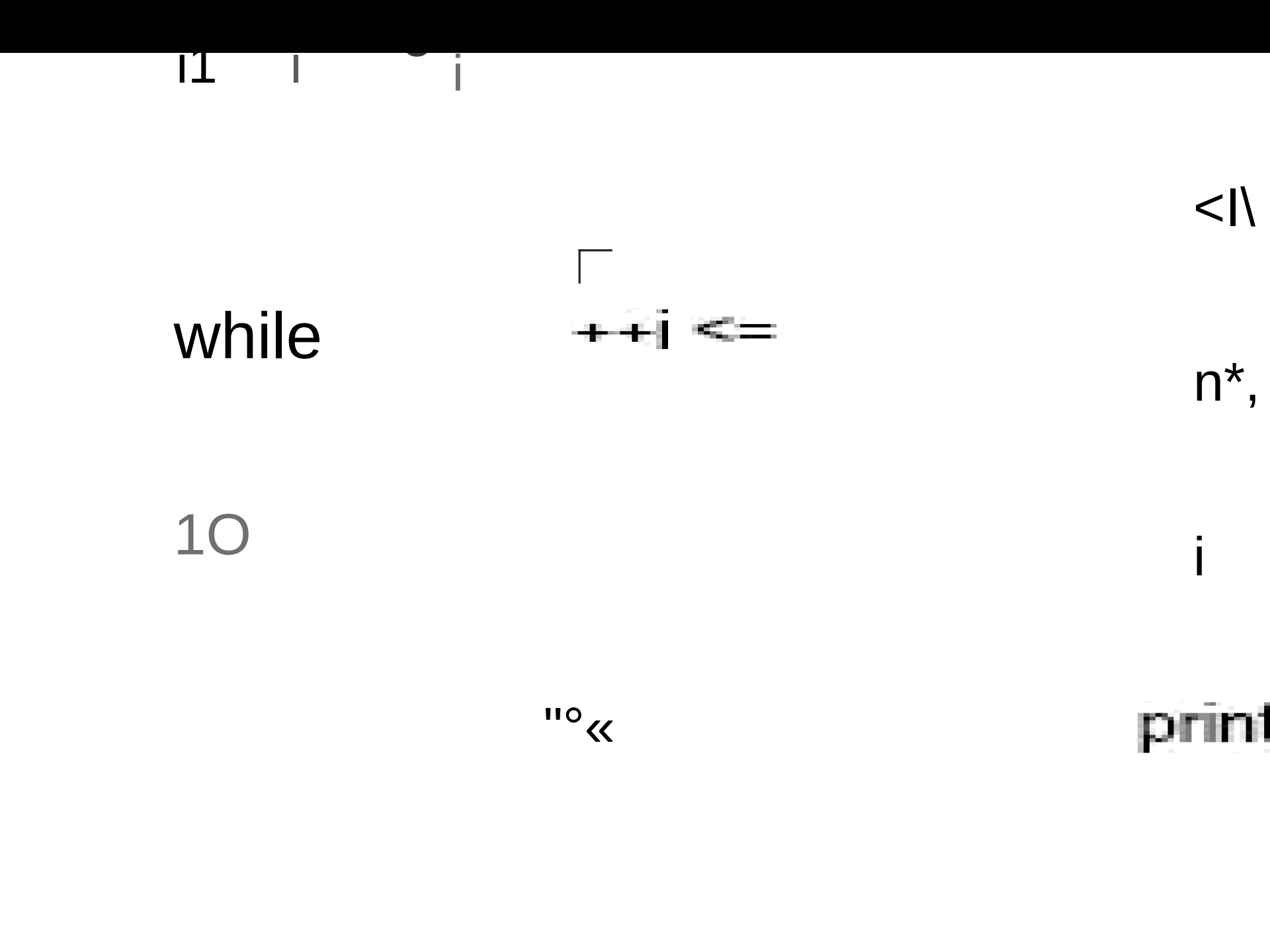
```
main(
L
```

```
int    i = 1 ;

vv1ile ( i  <= 1 D ) ;

└
        p rinN ( "°.éd\n". i
```

```
main(

        int
        while  1i  <=
                        0 1 )
                printf ( "%d\n", i ) ;
                   +=
                i          1
```

(e)

```
                                main(
```

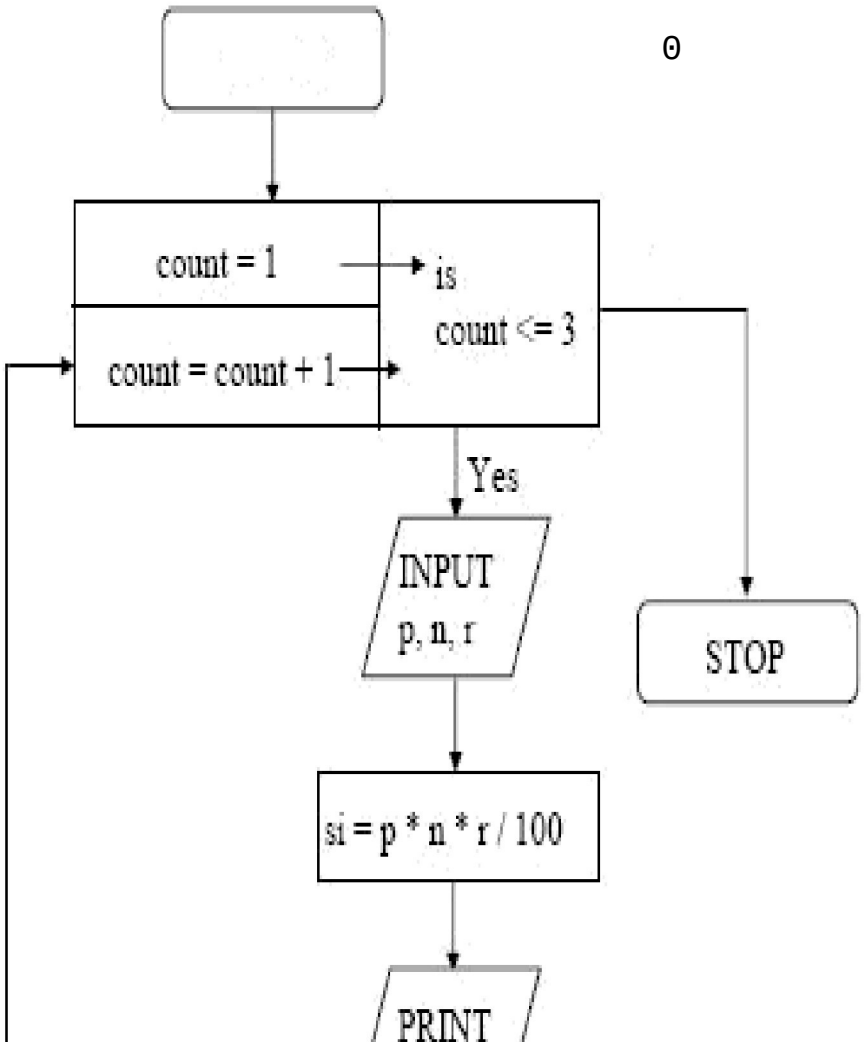

while
<I\
n*,
i
print

The *for* Loop

#(blCl/gDn0ttlN/9l3l@flbf/
59lS0l§,F âl¥lf /

START

```
main ( )
{
    int  p, n, count ;
    float  r, si ;

    for ( count = 1 ; count <= 3 ; count = count + 1 )
    {
        printf ( "Enter values of p, n, and r " ) ;
        scanf ( "%d %d %f", &p, &n, &r ) ;

        si = p * n * r / 100 ;
```



???? Is Valid ??????????????

???? What Will be the O/P ???????

i$?tiltiple Tiiitinñsntioiii in the J”or loop

Tlte iitirialisatiou expreaaiou of the foi• loop can coitiaiit more tltan DUe statezaent »eparatect b3•,' ?i CDII:?. Far exazaapEe,

fDr (i = 1, j = 2 j <= 10, j••)

*” Execution of n IOD|D SFI Ulfl known number Df times *

```
mnin( )

        char   anDthe,r

        int  nur,n

        do
```

```
printf ¿ "Enter a number ° )

scanf ¿ "°.écl". &num 'i

prinLf y "square of ’?°od is %é", num, num * num j
prinN ¿ "\n'A’nnt Lo enter another number y.’n " ) ;
scanf ¿ " 91‹c°, BanDther j ;
```

t• while ¿ another y' j

'"ddl«¿MsingJAiIeI«y’l

jlai arahei°';•

chaf arohef= ’/

,

prnf ı

"Erlefa

rvndef "j

") piirt ı ‹,

joy.jjlp erlefaralhe

sczrf \"°«R

frun#efy /r’¡

’,&nrr ¡

The *break* statement

```
main( )
{
        int num, i ;

        printf ( "Enter a number " ) ;
        scanf ( "%d", &num ) ;

        i = 2 ;
        while ( i <= num - 1 )
        {
                if ( num % i == 0 )
                {
                        printf ( "Not a prime number" ) ;
                        break ;
                }
                i++ ;
        }

         if ( i == num )
                printf ( "Prime number" ) ;
}
```

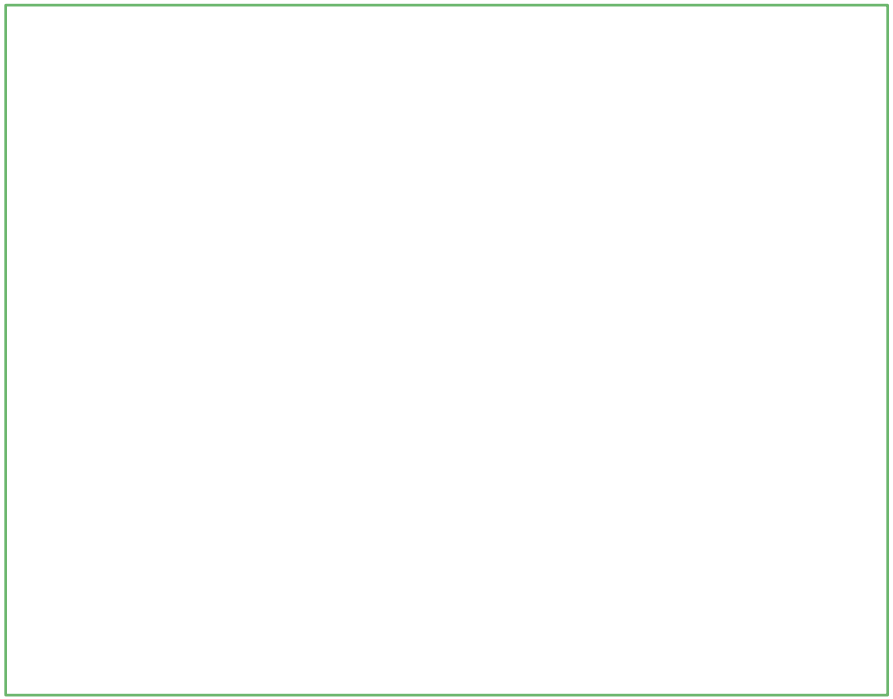
The *continue* statement

```
main( )
{
        int   i, j ;

        for ( i = 1 ; i <= 2 ; i++ )
        {
                for ( j = 1 ; j <= 2 ; j++ )
                {
                        if ( i == j )
                                continue ;

                        printf ( "\n%d %d\n", i, j ) ;
                }
        }
}
```

The *do – while* loop

Difference between *while* and *do-while*

- ***do-while*** **would execute its statements at least once, even if the condition fails for the first time.**
- **The *while*, on the other hand will not execute its statements if the condition fails for the first time.**



The Switch Case

```
switch ( integer expression )
{
        case constant 1 :
                do this ;
        case constant 2 :
                do this ;
        case constant 3 :
                do this ;
        default :
                do this ;
}
```

Consider the following program:

```
INI    1    2  ;
switch ( i )
        case 1
                printf ( °I am in case 1  \n° )      ;
        case 2
                printf ( °I am  in  case 2 \n° )     ,
        case 3
```

```
        printf ( °I am in case 3  \n° )      ;

default

        printf ( °I am in default \n° ) ,
```

The Correct solution is

```
main( )
{
      int   i = 2 ;

      switch ( i )
      {
            case 1 :
                  printf ( "I am in case 1 \n" ) ;
                  break ;
            case 2 :
                  printf ( "I am in case 2 \n" ) ;
                  break ;
            case 3 :
                  printf ( "I am in case 3 \n" ) ;
                  break ;
            default :
                  printf ( "I am in default \n" ) ;
      }
}
```

(a) The earlier propa am that xtsed s«4tch iizay gix•e you the vs•roiig iiizpressioii tliot you con use oxily cases orrakiped in ascending oxder. 1. ??. S end default. You can in feet put the cuses in any order yotl please. Here is aix exunzple of scxanibled cuse order:

```
main( )

    int  i    22 ,

    switch ( i )

        case 121

            pnn0 ( °I am in case 121 \n° )

            brea,k

        case 7

            prin0 ( °I am in case 7 \n° ) ,
            brea,k

        case 22

            prin0 ( °I am in case 22 \n° )
            break ,
```

```
default

        prin0 ( °I am in default \n° ) ,
```

You can use the character also

```
main()
{
    char c = 'x' ;

    switch ( c )
    {
        case 'v' :
            printf ( "I am in case v \n" ) ;
            break ;
        case 'a' :
            printf ( "I am in case a \n" ) ;
            break ;
        case 'x' :
            printf ( "I am in case x \n" ) ;
            break ;
        default :
            printf ( "I am in default \n" ) ;
    }
}
```

```
switch (

     ca
    case 'A' .
        prin0 ( °a as in ashar" ) ;
        break
    case 'b'
    case 'B' .
        prin0 ( "b as in brain° )

        break
    case 'c'
    case 'C' .
        prin0 ( °c as in cookie° ) ,

        break
    default
```

```
prin0 [ "wish you knew what are alphabets™ ) ,
```

@). We can check the value of any expression in a switch. Thus the fokowing switch statements are lepal.

```
swtch ( i + i
switch ( 23 • 45 % 4
* k ) swlch ( a < 4
&6 b» 7)
```

Expressions can also be used in cxses provided they are constant expressions. Thus case 3 + 7 is correct, however, case a + b is incorrect.

(i) The brenk statement when used in a switch takes the conbol outside &e switch. However, use of cnntinue w8l not take the conbol to the beginning of switch as one is likely to believe.

(j) In principle, a switch may occw within another, but in pmctice it is rarely done. Such statements would be caEed

nested switch statements.

GOTO statement

```
main( )
{
      int  goals ;

      printf ( "Enter the number of goals scored against India" ) ;
      scanf ( "%d", &goals ) ;

      if ( goals <= 5 )
            goto sos ;
      else
      {
            printf ( "About time soccer players learnt C\n" ) ;
            printf ( "and said goodbye! adieu! to soccer" ) ;
            exit( ) ;  /* terminates program execution */
      }

      sos :
            printf ( "To err is human!" ) ;
```

[A] What v'ould be the output of the follo r np

- (mai

```
charsuite =
3, switch
(suite )

        print(ñrDiamon n)

        print(1Spade°) ,

    default
```

```
print(ñnHeart°) ,

péntf (" \nI thought Dne wears a suite' ) ,
```

```
/* A menu driven
main( )

    int  choice ;
    while ( 1 )

        printf ( "\n1.  Factorial" ) ;
        printf ( "\n2.  Prime" ) ;

        switch (      hoi    )
            case 1
break ;                 /*                    facts rial elf  m turn ber          "
        printf ( "\n4.  Exit" ) ;Odd

                                  deciding prime number */break ;

        break ;
    case 4 :
        exit( ) ;
```