Date :...../...../.....

Question outline:

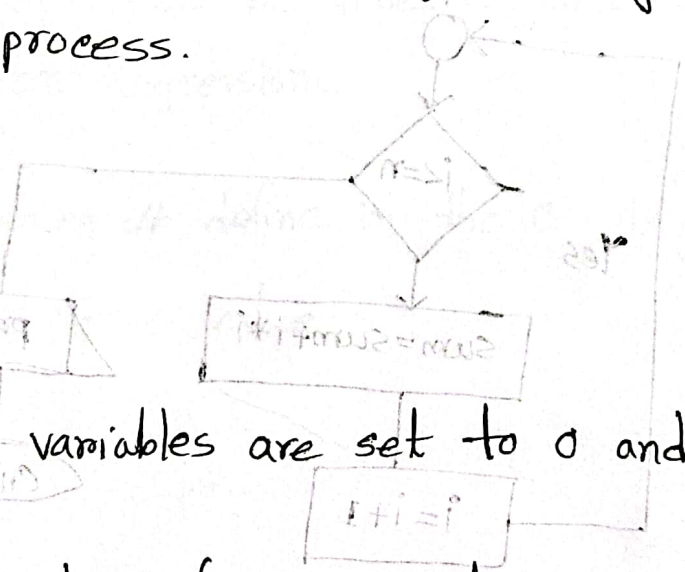
(chapter-1)

1. Write an algorithm and draw a flowchart to find out the sum of the following series:

$1^2+3^2+5^2+\cdots+N$ terms.

⇒ **Algorithm:** An algorithm is a procedure that used for solving a problem or performing a computation.

Flowchart: A flowchart is a graphical or symbolic representation of a process. A flowchart is a type of diagram that represents a workflow or process.



⇒ **Alt Algorithm:**

step-1: start

step-2: Sum and variables are set to 0 and 0 respectively take 1.

step-3: taking the value of n as input.

step 4: Compare the value of i with n. if i<=n, then go to step 6, else go to ~~spt~~ step 5

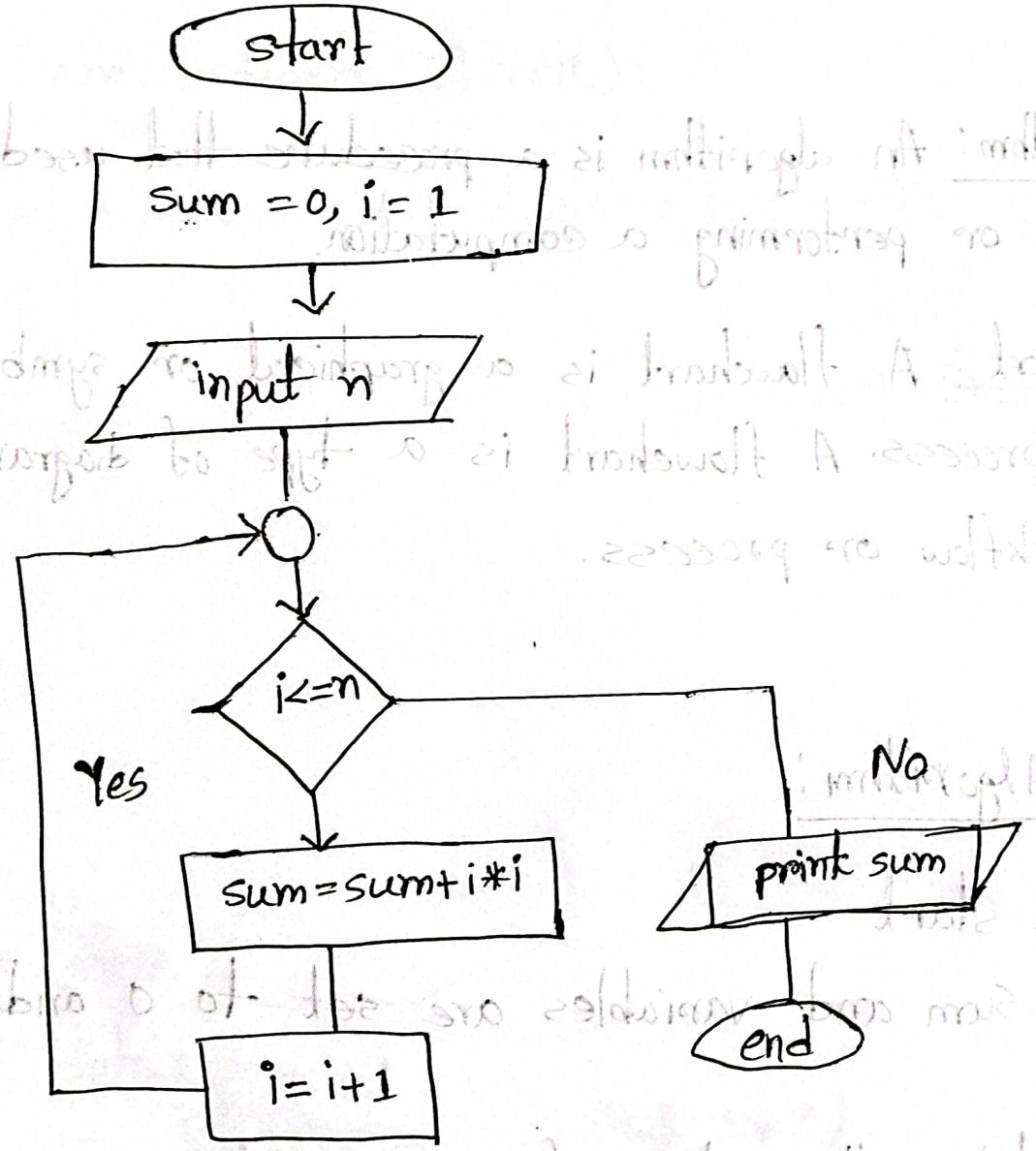
step-5: Sum variable value sum = sum + i*i and increment the variable value by 1.

Date :...../...../.....

step-6: Display the value of sum as the result.

step-7: Finish.

Flowchart:



2. Where & when do will use the #define directive?
why

⇒

In the C programming language the #define directive allows the definition of macros within our source code. These macro definations allows constant values to be declared for in using our code.

~~mare:~~

Macro difination are not variables and can't be changed by our program code like variable, we generally use this ~~symbol~~ syntax when creating constant that represent number, string or expression. We can use the #define directive to define a constant using an expression.

The syntax of using # define in the C language is,

#define constant_name value.

or

#define constant_name expression.

Here most C program define their constant name is upercase value contain the value of the constant. Here expression means whose value is assigned to the constant. The expression must be enclosed in program these if it contains operations ors

Example: #define Age 10 !

2023.03.25 23:06

#3. Why and when we used #include directive ?

⇒

#include is a way of including a standard or user-defined file in the program and is mostly written at the beginning of any c/c++ program.

#include is a processor directive that is used for file inclusion in a c-program. #include is also known as a file inclusion directive.

#include directive is used to add the content/ piece of code from a reserved header file into our code file before the compilation of our c program. These header files include definition of many predefined function like printf(), scanf(), getch() etc.

The sytax for #include directive in the c language is

#include <header file>

Here the header file that we with to include. A headerfile is a file that tripically ends in ".h".

Example: #include <stdio.h>

```
int main () {
int a=10, b=20;
 int sum = a+b;
 printf ("the result is %d", sum);
 return 0; }
```

Bichitra

2023.03.25 23:06

Date :...../...../.....

In this program we are using the #include directive.

3. What does void main(void) mean?

⇒

The void main() indicated that main() function will not ~~refers~~ returns any value. When our ~~prgr~~ program is simple, and it is not going to terminale before reching the last line of the code, or the code is error free, then we can use the void main(),

4. Why do we need to use comments in programs?

⇒

A comment is an explanation or decription of the source code of a program. It helps a developer explain logic of the code and improves program read ability. Proper use of commenting can make code maintance much ~~easier~~ easier as well as helping make finding bugs faster.

Comments may be multilines or a single line, depeding upon what we need. Single line comments are started with double slash. Ex. // this is a sigle line comment.

Multilines comments are started with slash asterisk (/*) and ends with a asterisk slash (*/). It can be written anywhere in a program. Ex. /* This is a multiline comments */

5. Describe the structure of a C program.

⇒ To write a C-program, we first create functions and then put them together. A C program may contain one or more sections as shown below:

Documentation section:

The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later. ~~The de~~

Link section: The link section provides instructions to the compiler to link functions from the system library.

Definition section: The definition section defines all symbolic constants.

Global declaration section: There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also ~~declared~~ declares all the user-defined functions.

Bidisha

Date :...../...../.....

Main() function section: Every C program must have one main() function section. This section contains two parts, declaration part and executable part. The declaration part declares all the variables used in the excutable part. There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. All parts ends with a semicolon (;)

Subprogram section: The subprogram section contains all the user defined functions that are called in the main function

.36) 6. (2.1) Find errors, if any, in the following program :

```
/* A simple program
  int main()
  {
    /* Does nothing */
  }
```

⇒

Bidita

Date :...../...../.....

p-65)

6. Describe the four basic data types. How could we extend the range of values they represent?

⇒ The four basic data types is

(I) Integer
(II) character
(III) Float
(IV) Void

I) Integer type: Integers are whole numbers with a range of values supported by a particular machine. Generally, integers occupy one word of storage, and since the word sizes of machines vary (typically, 16 or 32 bits) the size of an integer that can be stored depends on the computer. A signed integer uses one bit for sign and 15 bits for the magnitude of the number. C has three classes of integer storage, namely short int, int and long int, in both signed and unsigned forms.

General form: int <variable name>;
int num1;

II) Character types: A single character can be defined as a character (char) type data. Characters are usually stored in 8 bits (one byte) of internal storage. The quantifier signed or unsigned may be explicity applied to char. While unsigned chars have values between 0 and 225, signed char have values from -128 to 127.

2023.03.25 23:06

General form: char<variable name>; char ch = 'a';

Bidisha

III) Floating point types: Floating point numbers are stored in 32 bits, with 6 digits of precision. Floating point numbers are defined in C by the keyword float. When the accuracy provided by a float number is not sufficient, the type double can be used to define the number. A double data type number uses 64 bits giving a precision of 14 digits.

General form: float <variable name>;

float num1;

IV) Void types: The void types has no value. This is usually used to specify the type of functions. The type of a function is said to be void when it does not return any value to the calling function.

We can use the short, long, signed and unsigned keywords to extend the primary data types. A short data type has a smaller range compared to the long data type.

Date :...../...../.....

(3.7)

7. What is a variable and what is meant by the 'value' of a variable?

⇒ A variable is a named unit of data that is assigned a value. If the value ~~that~~ is modified, the name does not change. ~~Variables are used with~~ Variable is bacically nothing but the name of a memory location that we use for storing data. We can change the value of a valable in c or any other language, and we can also reuse it multiple times. We ~~ee~~ use symbols in variables for representing the memory location - so that it becomes easily identifiable by any user.

Bichitra

Date :...../...../.....

(3.8) 8. How do variables and symbolic names differ?

⇒ Variables and symbolic names are both used to represent values in programm and computer science, but they differ in their usage and meaning.

A variable is a storage location in a computers memory that holds a value. In programming, a variable is typically declared with a specific data type and assigned a value during runtime. The value of a variable can be changed during the ~~exeu~~ execution of a program, and it can be used to perform calculations, comparisons and other operations.

A symbolic name, on the other hand, is a name given to represent a value or concept in code. It is a label that is used to identify a particular value or object, but it does not hold any actual value itself. Symbolic name are typically used to make code more readable and maintainable, by providing a meaningful name that describes the purpose or function of a particular value or object.

For example, in a program that calculates the area of a circle, a variable might be declared as 'radius' and assigned a value of 5.0. The symbolic name 'PI' might also be defined in the code to represent the mathematical constant Pi, which is used in the

Bichitra

Date :...../...../.....

9. What is operator in C language ?

⇒

Operator: An operator is a symbol that operates on a value or a variable. For example: '+' is an operator to perform addition. C has a wide range of operators to perform ~~variabl~~ various operations. There are eight types of operator [~~describing~~ described before] [for CT].

10. How many number of categories in C language ~~of~~ operators.

⇒ 8 categories (describe) (details).

⇒ [described before].

11. Explain the comma operator & sizeof operator with example

⇒ They are special type operator..

The comma operator:

The comma operator can be used to link the related expressions together. A comma-linked list of ~~opera~~ expressions are evaluted left to right and the value of right-most expression is the value of of the combined expression. For example:

the statement, value = (x=10, y=5, x+y);

First assigns the value 10 to x, the assigned 5 to y,

Date : ... / ... / ...

and finally assigns 15 (i.e. 10+5) to value. Some applications of comma operators are:

In for loops:

```
for(n=1, m=10; n<=m; n++, m++)
```

In while loops:

```
while (a=getchar(), a != '10')
```

Exchanging values:

```
t=x, x=y, y=t;
```

The size-of operator:

The sizeof operator is a compile time operator and, when used with an operand, it returns the number of bytes the operand occupies. The ~~ope~~ operend may be a variable, a constant or a data type qualifier.

Examples:

```
m= sizeof (sum);
n= sizeof (long int);
k= sizeof (235L);
```

The sizeof operator is normally used to determine the lengths of arrays and structures when their sizes

Bidhitra

Date :...../...../.....

are not known to the programmer. It is also used to allocate memory space dynamically to variables during execution of a program.

11. What do you mean by type conversion in expression?

⇒

In programming, Type conversion in expressions refers to the automatic or explicit conversion of one data type to another data type when they are used together in an expression.

For example, suppose we have an expression that involves both an integer and a floating-point number, like '5+2.5'. In this case, the integer 5 will be automatically converted to a floating-point number before the addition operation is performed, resulting in the expression evaluating to '7.5'.

There are two type of conversions: Implicit and explicit. Implicit conversions are performed autonomically by the programming language, while explicit conversions are performed explicitly through the use of type-casting operations.

Type conversions can be important in programming because they can affect the accuracy of calculations, as well as the

Date :...../...../.....

efficiency and readability of code. Programmers need to be aware of the rules governing type conversions in their programming language to ensure that their code behaves as expected.

12. Differentiate between implicit and explicit type conversions.

⇒