Computer Programming & Problem Solving
CS100

Mrs Sanga G. Chaki
Department of Computer Science and Engineering
National Institute of Technology, Goa
March, 2023

Introduction

1. What is a problem?
2. Steps of Problem Solving?
    1. Understand the problem
    2. Devise a plan
    3. Carry out the plan (action)
    4. Look back and check
3. What is a program?
    1. Set of instructions to a computer to achieve some goal

Introduction

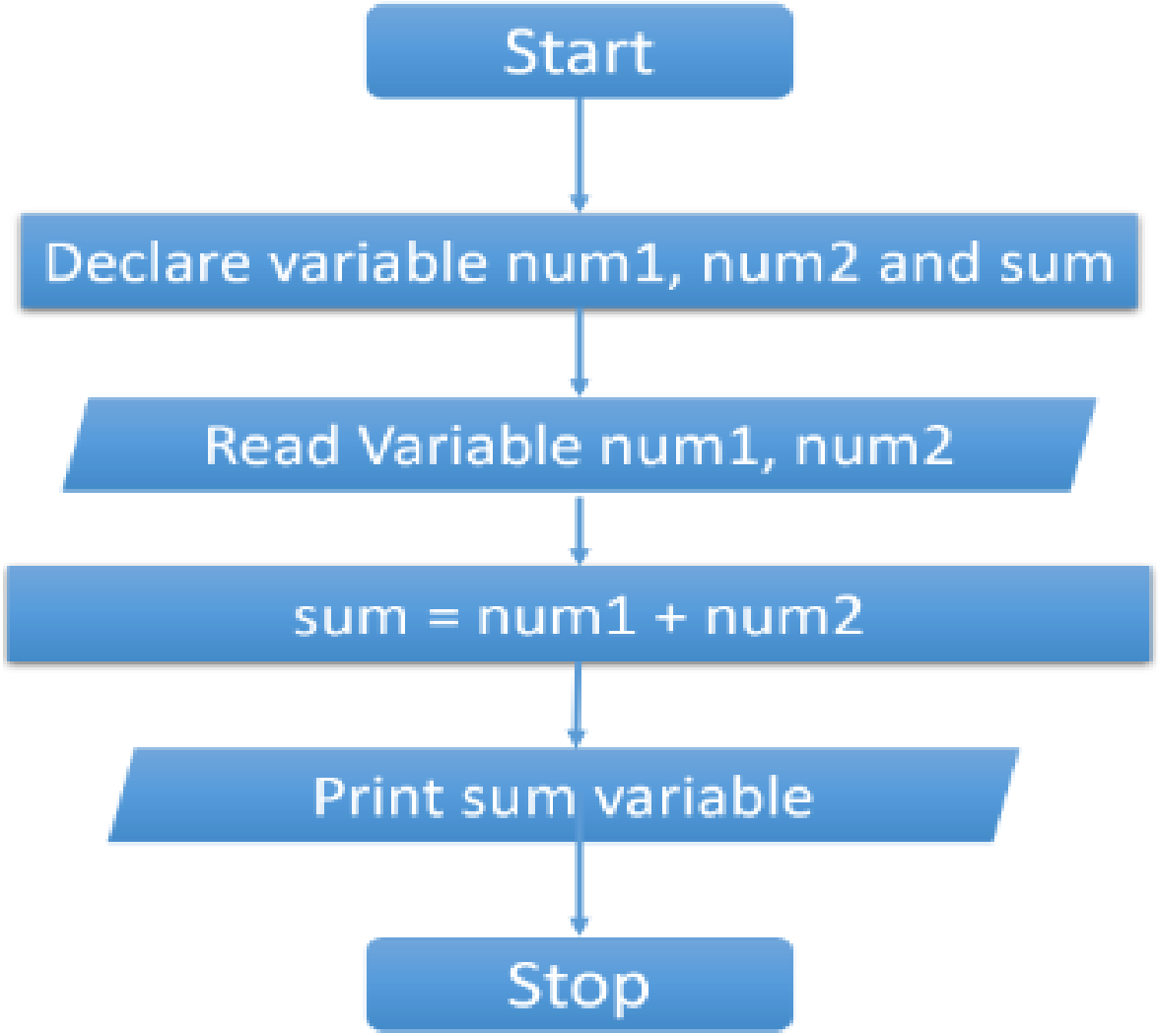
1. How are these related?
    1. A program is a list of instructions composed in such a way as to enable a computer to solve a problem, broken down into successively smaller parts.
2. Can all problems be solved by programming?
    1. No, eg. Non-computable problems

First step to Problem Solving

1. Algorithm
   a) In mathematics and computer science,
   b) finite sequence of rigorous instructions
   c) typically used to solve a class of specific problems
   d) or to perform a computation.

2. Flowchart
   a) A type of diagram that represents a workflow or process.
   b) A diagrammatic representation of an algorithm
   c) A step-by-step approach to solving a task.
3. The order of instructions is important in algos and flowcharts

Mathematical problem – Algo to Add two Numbers



Mathematical problem – Add two Numbers - Code

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int num1, num2, sum;
    printf("Please enter 2 integer numbers: ");
    scanf("%d %d", &num1, &num2);
    sum = num1 + num2;
    printf("The sum = %d", sum);
    return 0;
}
```

1. **Start instructions**
2. **Declaring variables**
3. **Output**
4. **Input**
5. **Arithmetic instruction**
6. **Stop instruction**

Mathematical problem – Add two Numbers - Output

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int num1, num2, sum;
    printf("Please enter 2 integer numbers: ");
    scanf("%d %d", &num1, &num2);
    sum = num1 + num2;
    printf("The sum = %d", sum);
    return 0;
}
```

```
Please enter 2 integer numbers: 10 20
The sum = 30
```

The C Language

1. What is C?
2. Some rules to follow when writing C code
3. Important aspects of C
    1. C character set, operators, C variables, naming convention, C keywords
    2. C instructions – types
        a) Type Declaration Instruction
        b) Arithmetic Instruction
            - Integer and Float Conversions, Hierarchy of Operations, Associativity of Operators
        c) Control Instructions
            - Sequence Control Instruction, Selection or Decision Control Instruction, Repetition or Loop Control Instruction, Case Control Instruction
        d) Input and output instructions

What is C?

1. A general purpose programming language
2. Used in operating systems, device drivers, protocol stacks, computer architectures of super computers to microcontrollers
3. Designed by: Dennis Ritchie almost 50 years ago
4. Successor to language B

Why do we need C?

1. Computers do not understand English, they understand Binary
2. Humans are more comfortable with natural languages
3. Programming languages bridge this gap
4. Human to human → Natural languages
5. Human to computers → Programming languages

Example code

```c
// Online C compiler to run C program online
#include <stdio.h>

int main() {
    int num1, num2, sum;
    printf("Please enter 2 integer numbers: ");
    scanf("%d %d", &num1, &num2);
    sum = num1 + num2;
    printf("The sum = %d", sum);
    return 0;
}
```

General Rules to write C code

1. Each instruction in a C program is written as a separate statement.
2. The statements in a program must appear in the same order in which we wish them to be executed – like in algorithm
3. No unnecessary blank spaces are allowed
4. All statements are entered in small case letters
5. Every C statement must end with a ;.
6. Comment about the program should be enclosed within /* */ or //.
7. main( ) is a collective name given to a set of statements. This name has to be main( ), it cannot be anything else. All statements that belong to main( ) are enclosed within a pair of braces { }

General Rules to write C code

1. main( ) is a function. Every function has a pair of parentheses ( ) associated with it. printf(), scanf(), pow(), sqrt() are also functions.
2. Any variable used in the program must be declared before using it.
3. The general form of printf( ) function is,
   a) printf ("<format string>", <list of variables>) ;
   b) <format string> can contain, %f for printing real values, %d for printing integer values, %c for printing characters.
   c) Eg. printf ("Prin = %d \nRate = %f", p, r) ;
4. The general form of scanf( ) function is,
   a) scanf ("<format string>", <address of variables>) ;
   b) That's why we use the &, which indicates address.

More rules

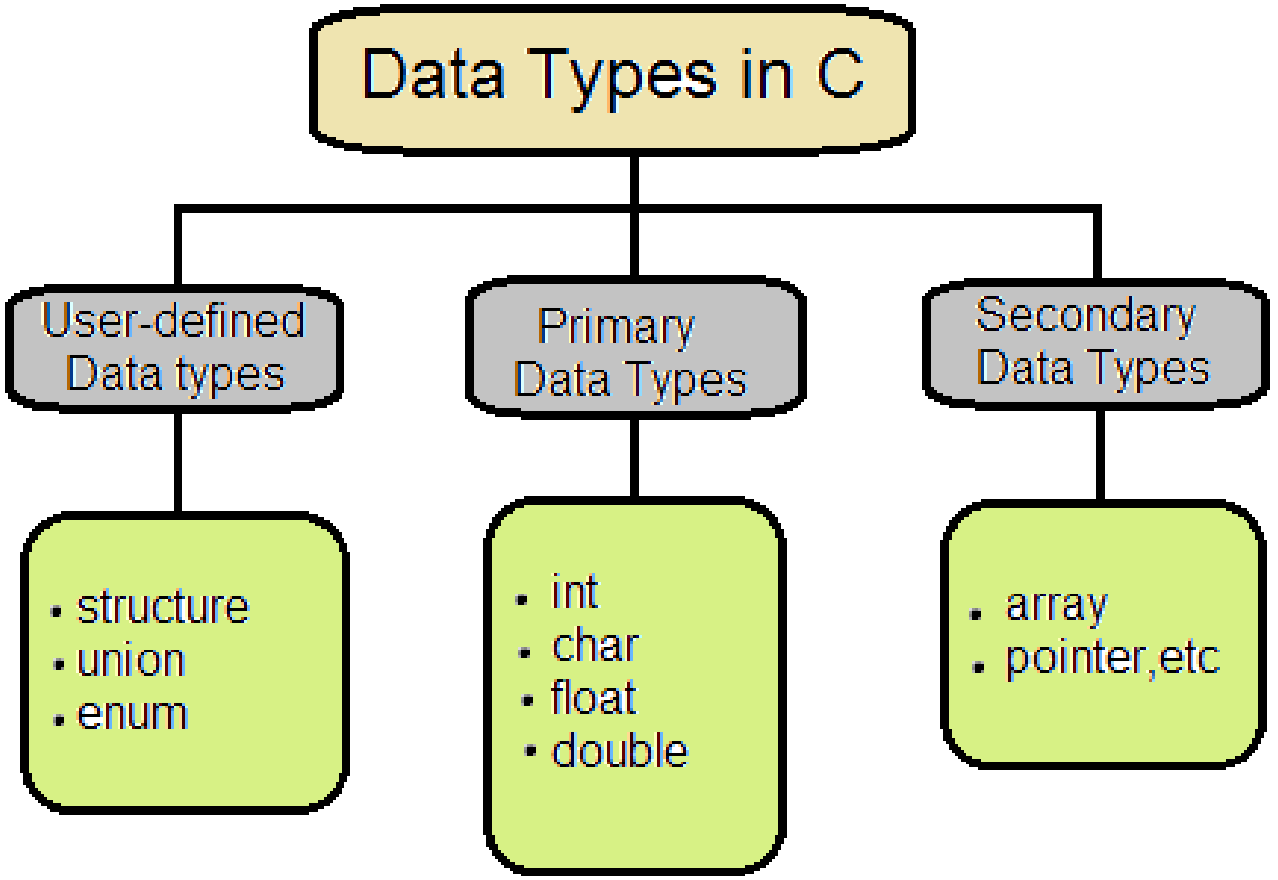
1. #include<stdio.h> and #include<math.h>
   a) stdio.h and math.h → header file.
   b) They contain important code already written so that we can use printf(), scanf(), pow(), sqrt() etc.
   c) #include → Instruction to add this file to my code. So I am including this already written code in my code.
2. The other two things are int main() and return 0;
3. We will discuss their purpose at a later stage

C character set and operators

1. A character denotes any alphabet, digit or special symbol used to represent information.
2. Using these characters we create meaningful data. What types of data can we have?

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

C Data Types



1. A particular kind of data item
2. Defines the values it can take and
3. The operations that can be performed on it.

C Constants and Variables

1. A constant is an entity that doesn’t change
2. A variable is an entity that may change
3. They can be of any datatype described in the previous slide.

C Variables – Rules to construct names

1. A variable name is any combination of 1 to 31 alphabets, digits or underscores
2. Do not create unnecessarily long variable names
3. Create problem related meaningful variable names
4. The first character must be an alphabet or underscore.
5. No commas or blanks are allowed.
6. No special symbol other than an underscore can be used
7. It is compulsory to declare the datatype of any variable

C Keywords

1. Keywords are the words whose meaning has already been explained to the C compiler.
2. The keywords cannot be used as variable names

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

Type Declaration Instruction

1. Used to declare the type of variables used in a C program
2. Any variable used in the program must be declared before using it in any statement.
3. The type declaration statements are generally written at the beginning of main( ) function.
4. Eg.
    1. int num1;
    2. float quotient ;
    3. char grade;

Arithmetic Instructions - Rules

1. A C arithmetic instruction consists of one variable name on the left hand side of =
2. And variable names & constants on the right hand side of =.
3. The variables and constants appearing on the right hand side of = are connected by arithmetic operators like +, -, *, and /.
4. That's why we cannot write **num1+num2=sum;**
5. How the execution of an arithmetic statement takes place?
   a) Firstly, the right hand side is evaluated using constants and the numerical values stored in the variable names.
   b) This value is then assigned to the variable on the left-hand side.

Arithmetic Instructions - Rules

1. C allows only one variable on left-hand side of =. That is, z = k * l is legal, whereas k * l = z is illegal.
2. No operator is assumed to be present.
   a) a = c.d.b(xy) usual arithmetic statement
   b) b = c * d * b * (x * y) C statement
3. Unlike other high level languages, there is no operator for performing exponentiation operation – use pow()
4. Here pow( ) function is a standard library function.

```
#include <math.h>
main( )
{
    int a ;
    a = pow ( 3, 2 ) ;
    printf ( "%d", a ) ;
}
```

Arithmetic Instructions

1. A C arithmetic statement could be of three types
   a) Integer mode arithmetic statement - This is an arithmetic statement in which all operands are either integer variables or integer constants.
      - Rule: An arithmetic operation between an integer and integer always yields an integer result
   b) Real mode arithmetic statement - This is an arithmetic statement in which all operands are either real constants or real variables.
      - Rules: An operation between a real and real always yields a real result.
   c) Mixed mode arithmetic statement - This is an arithmetic statement in which some of the operands are integers and some of the operands are real
      - Rules: An operation between an integer and real always yields a real result. In this operation the integer is first promoted to a real and then the operation is performed. Hence the result is real.

Integer and Float Conversions - Rules

1. An arithmetic operation between an integer and integer always yields an integer result
2. An operation between a real and real always yields a real result.
3. An operation between an integer and real always yields a real result. In this operation the integer is first promoted to a real and then the operation is performed. Hence the result is real.

Operation	Result	Operation	Result
5 / 2	2	2 / 5	0
5.0 / 2	2.5	2.0 / 5	0.4
5 / 2.0	2.5	2 / 5.0	0.4
5.0 / 2.0	2.5	2.0 / 5.0	0.4

Hierarchy and Associativity of Operations

1. The priority or precedence in which the operations in an arithmetic statement are performed is called the hierarchy of operations.

Priority	Operators	Description
1st	* / %	multiplication, division, modular division
2nd	+ -	addition, subtraction
3rd	=	assignment

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

operation: /
operation: *
operation: /
operation: +
operation: +