2.2 Primitive Data Types

**Primitive Data Types:**

There are eight primitive data types supported by Java. Primitive data types are predefined by the language and named by a keyword.

**Eight primitive data types**

**Byte:**

Byte data type is an 8-bit signed two's complement integer.

Default value : 0

Byte data type is used to save space in large arrays mainly in place of the integers, since a byte is four times smaller than an int.

Example: byte a = 100 , byte b = -50

**Short:**

Short data type is a 16-bit signed two's complement integer.

Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an int.

Default value is 0.

Example: short s = 10000, short r = -20000

**Int:**

Int data type is a 32-bit signed two's complement integer.

Int is usually used as default data type for integral values unless there is a concern about memory.

The default value : 0.

Example: int a = 100000, int b = -200000

**Long:**

Long data type is a 64-bit signed two's complement integer.

Example: long a = 100000L, int b = -200000L

**Float:**

Float data type is a single-precision 32-bit IEEE 754 floating point.

Float is mainly used to save memory in large arrays of floating point numbers.

Default value : 0.0f.

Float data type is never used for precise values like currency.

Example: float f1 = 234.5f

**Double:**

Double data type is a double-precision 64-bit IEEE 754 floating point.

This data type is generally used as the default data type for the decimal values, usually the default choice.

Double data type should never be used for precise values such as currency.

Default value is 0.0d.

Example: double d1 = 123.4

**Boolean:**

Boolean data type represents one bit of information.

There are two possible values: true and false.

This data type is used for simple flags which track true/false conditions.

The default value is false.

Example: boolean one = true

**Char:**

char data type is a single 16-bit Unicode character.

Minimum value : '\u0000' (or 0).

Maximum value : '\uffff' (or 65,535 inclusive).

Char data type is used for storing any character.

Example: char letterA ='A'

**Reference Data Types:**

Reference variables are created using defined the constructors of classes. They are used to access objects. These variables are declared to be of a specific type which cannot be changed. For example, Puppy, Employee, etc.

Class objects and various types of array variables comes under reference data type.

Default value of any reference variable is null.

A reference variable can be used to refer any object of declared type or any compatible type.

Example: Animal animal = new Animal("giraffe");

**Java Literals:**

A literal is a source code representation of a fixed value.

It is represented directly in the code without any computation.

Literals can be assigned to any primitive type variable.

For example:

byte a = 68;char a = 'A'

byte a = 68;

char a = 'A'.

2.2.1 Strings

**Let us see any two library functions used for string handling.**

i) strcat()

ii) strcmp()

iii) strrev()

**Let us discuss about the string representation in C language.**

Strings are represented using array of type char.

**Let us see about the string.**

String is a array of characters.

**The Standard Files**

The C programming treats all the devices as files. Therefore the devices like the display are being addressed in the same way as the files and the following three files are automatically opened when a program is being executed to give access to the keyboard and screen.

|  |  |  |
| --- | --- | --- |
| **Standard File** | **File Pointer** | **Device** |
| Standard input | stdin | Keyboard |
| Standard output | stdout | Screen |
| Standard error | stderr | Our screen |

The file pointers are the means to access the file for reading and writing purpose.

**The getchar() and putchar() Functions**

The int getchar(void) function reads the next available character from the screen and returns as an integer. This function reads only single character at a time. This method can be used in the loop, in case if we want to read more than one character from the screen.

The int putchar(int c) function puts the passed character on the screen and it returns the same character. This function puts only a single character at a time. This method can be used in the loop, in case if we want to display more than one character on the screen.

**Example:**

#include <stdio.h>int main( ) {

int c;

printf( "Enter a value :"); c = getchar( );

printf( "\nYou entered: "); putchar( c );

return 0;

}

When above code is compiled and executed, it waits for us to input some text. When we enter a text and press enter, then program proceeds and reads only a single character and displays it as follows ,

$./a.out

Enter a value : Content Hidden

You entered: C

**The gets() and puts() Functions**

The char \*gets(char \*s) function will read a line from stdin into buffer being pointed to s until either a terminating newline or EOF (End of File).

The int puts(const char \*s) function will write the string 's' and 'a' trailing newline to stdout.

**Example:**

#include <stdio.h>int main( ) {

char str[100];

printf( "Enter a value :"); gets( str );

printf( "\nWe entered: "); puts( str );

return 0;

}

When the above code is compiled and executed, it waits for us to input some text. When we enter a text and press enter, then the program shall proceed and read the complete line till end and displays it as follows ,

$./a.out

Enter a value : this is test

We entered: This is test

**The scanf() and printf() Functions**

The int scanf(const char \*format, ...) function reads the input from standard input stream stdin and it scans the input according to format provided.

The int printf(const char \*format, ...) function writes the output to the standard output stream stdout and it produces the output according to format provided.

The format will be a simple constant string, but we can specify %s, %d, %c, %f, etc., to print or read the strings, integer, character or float respectively. There are many other formatting options available which will be used based on requirements.

**Example:**

#include <stdio.h>int main( ) {

char str[100]; int i;

printf( "Enter a value :"); scanf("%s %d", str, &i);

printf( "\nWe entered: %s %d ", str, i);

return 0;

}

When the above code is compiled and executed, it waits for us to input some text. When we enter a text and press enter, then program proceeds and reads the input and displays it as follows ,

$./a.out

Enter a value : seven 7

We entered: seven 7

Programs

**Here we discuss about a C program that counts number of words in a given string.**

#include<stdio.h>

#include<conio.h>

void main()

{

char s[50], ch;

int i, c = 0;

clrscr();

printf("Enter any string:");

for(i=0; ch! = '\0'; i++)

{

ch = getchar();

s[i] = ch;

}

s[i] ='\0';

for(i=0; s[i]! = '\0'; i++)

{

if(s[i] = -’ ‘)

{

C**++;**

while (s[i] = = ")

i++;

}

}

printf("\n\n Total words are %d", c+1);

getch();

}

**Let us discuss to write a program to find number of vowels and consonants in a text string.**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char str[30];

int vow = 0, cons = 0, i = 0;

clrscr ();

printf("Enter a string:");

void main ()

{

char str[30];

int vow = 0, cons = 0, i = 0;

clrscr ();

printf("Enter a string:");

gets(str);

while(str[i]!='\0')

{

if(str [i] = = 'a' || str [i] = ='A' || str[i] = ='c' || str[i] = = 'E' || str [i] = = 'i' || str [i] = = 'I' || strf[i] = = '0' || str [i] = = '0' || str[i] = = 'u' || str[i] = = 'U')

vow + +;

else

cons + +;

i + +;

}

printf("\n Number of Vowels = ***%*** d'; vow);

printf("\n Number of Consonants = % d'; cons);

getch ();}

**String Declaring, Initializing, Printing and reading strings**

In C programming, array of character are called strings. A string is terminated by the null character /0. Example:

"c string office"

Here, "c string tutorial" is a string. When, compiler encounters strings, it appends null character at the end of string. Memory diagram of strings in C programming.



**Declaration of strings:**

Strings are declared in C in similar manner as the arrays. Only difference is that, strings are of char type.

char s[5];



Strings can also be declared using the pointer.

char \*p

**Initialization of strings:**

In C, string could be initialized in different number of ways.

char c[]="abcd";

OR,

char c[5]="abcd";

OR,

char c[]={'a','b','c','d','\0'};

OR;

char c[5]={'a','b','c','d','\0'};

**Initialization of strings in C programming:**



String can also be initialized using the pointers.

char \*c="abcd";

Reading Strings from user.

Reading words from user.

char c[20];

scanf("%s",c);

String variable c can only take a word. It is because when white space is encountered, the scanf() function terminates.

**C program to illustrate how to read string from terminal:**

#include <stdio.h>

int main(){

char name[20];

printf("Enter name: ");

scanf("%s",name);

printf("Your name is %s.",name);

return 0;

}

**Output**

Enter name: Narendra Modi

Your name is Narendra.

Here, program will ignore Modi because, scanf() function takes only string before white space.

**Reading a line of text:**

C program to read the line of text manually.

#include <stdio.h>

int main(){

char name[30],ch;

int i=0;

printf("Enter name: ");

while(ch!='\n') // terminates if user hit enter

{

ch=getchar();

name[i]=ch;

i++;

}

name[i]='\0'; // inserting null character at the end

printf("Name: %s",name);

return 0;

}

This process to take the string is tedious. There are predefined functions such as gets() and puts() in C language to read and display the string respectively.

int main(){

char name[30];

printf("Enter name: ");

gets(name); //Function to read string from user.

printf("Name: ");

puts(name); //Function to display string.

return 0;

}

Both, above program has same output below:

**Output:**

Enter name: James Reed

Name: James Reed

**Passing Strings to Functions:**

String could be passed to function in similar manner as arrays as, string is also an array.

#include <stdio.h>

void Display(char ch[]);

int main(){

char c[50];

printf("Enter string: ");

gets(c);

Display(c); // Passing string c to function.

return 0;

}

void Display(char ch[]){

printf("String Output: ");

puts(ch);

}

Here, string c is passed from the main() function to user-defined function Display(). In function declaration, ch[] is the formal argument.

**String handling functions:**

We can perform the different type of string operations manually like finding the length of string, concatenating two strings etc. But, for programmers ease, many library function are defined under header file <string.h> to handle these commonly used talk in the C programming.

**Let us see any two library functions used for string handling.**

i) strcat()

ii) strcmp()

iii) strrev()

**Let us discuss about the string representation in C language.**

Strings are represented using array of type char.

**Let us see about the string.**

String is a array of characters.

**Programs:**

**Here we discuss about a C program that counts number of words in a given string.**

#include<stdio.h>

#include<conio.h>

void main()

{

char s[50], ch;

int i, c = 0;

clrscr();

printf("Enter any string:");

for(i=0; ch! = '\0'; i++)

{

ch = getchar();

s[i] = ch;

}

s[i] ='\0';

for(i=0; s[i]! = '\0'; i++)

{

if(s[i] = -’ ‘)

{

C**++;**

while (s[i] = = ")

i++;

}

}

printf("\n\n Total words are %d", c+1);

getch();

}

**A program to find the number of vowels and consonants in a text string.**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

char str[30];

int vow = 0, cons = 0, i = 0;

clrscr ();

printf("Enter a string:");

void main ()

{

char str[30];

int vow = 0, cons = 0, i = 0;

clrscr ();

printf("Enter a string:");

gets(str);

while(str[i]!='\0')

{

if(str [i] = = 'a' || str [i] = ='A' || str[i] = ='c' || str[i] = = 'E' || str [i] = = 'i' || str [i] = = 'I' || strf[i] = = '0' || str [i] = = '0' || str[i] = = 'u' || str[i] = = 'U')

vow + +;

else

cons + +;

i + +;

}

printf("\n Number of Vowels = ***%*** d'; vow);

printf("\n Number of Consonants = % d'; cons);

getch ();}

This program print a string. String could be printed by using various functions such as printf, puts.

**The C programming code:**

#include <stdio.h>

int main()

{

char array[20] = "Hello World";

printf("%s\n",array);

return 0;

}

To input a string we use the scanf function.

**C programming code:**

#include <stdio.h>

int main()

{

char array[100];

printf("Enter a string\n");

scanf("%s", array);

printf("You entered a string %s\n",array);

return 0;

}

Input string containing spaces.

#include <stdio.h>

int main()

{

char a[80];

gets(a);

printf("%s\n", a);

return 0;

}

Note that scanf could only input single word strings to receive strings containing spaces use gets function.

**C program to print the string using recursion:**

#include <stdio.h>

void print(char\*);

int main() {

char s[100];

gets(s);

print(s);

return 0;

}

void print(char \*t) {

if (\*t == '\0')

return;

printf("%c", \*t);

print(++t);

}

**Print string using loop:**

We print the string using for loop by printing individual characters of string.

#include <stdio.h>

#include <string.h>

int main() {

char s[100];

int c, l;

gets(s);

l = strlen(s);

for (c = 0; c < l; c++)

printf("%c", s[c]);

return 0;

}

**Reading String with spaces by using scanf():**

Scanf with %s accepts only String which does not contain white spaces (blanks/space).

C Supports the special edit set conversion code, by using this method we can accept the line of String with spaces using scanf().

**Syntax :**

scanf("%[\^n]", name );

Live Example :

char name[100];

printf("\nEnter the name : ");

scanf("%[\^n]",name);

printf ("\nName of Student : %s ",name);

**Output of this Block:**

Enter the Name : Yogi B

Name of Student :Yogi B

2.2.2 Array Types

**Definition**

We can initialize array in C either one by one or by using a single statement as follows:

double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};

The number of values between the braces { } cannot be larger than number of elements that we declare for the array between square brackets [ ]. An example to assign a single element of the array:

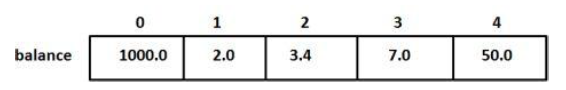
If we omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if we write

double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};

We will create exactly the same array.

balance[4] = 50.0;

The above statement assigns element number 5th in the array a value of 50.0. Array with 4th index will be 5th i.e. last element because all arrays have 0 as the index of their first element which is also called base index. Following is the pictorial representation of the same array:



**One dimensional and Two dimensional arrays**

**Let us discuss the example code to declare two dimensional array.**

type array\_name [row\_site] [column\_size];

int table [2] [3];

int table [2] [3] = {{0,0,0}, {1,1,1}};

**Here we shall see what happens when we access the array more than its dimension.**

When we access the array more than its dimension some garbage value is stored in the array.

**Multidimensional arrays and array applications:**

C programming language allows multidimensional arrays. Here is the general form of a multidimensional array declaration:

type name[size1][size2]...[sizeN];

For example, following declaration creates a three dimensional 5 . 10 . 4 integer array:

int threedim[5][10][4];

**Example:**

**A C program to find sum of two matrix of order 2\*2 using multidimensional arrays where, elements of matrix are entered by the user.**

#include <stdio.h>int main()

{

float a[2][2], b[2][2], c[2][2];

int i,j;

printf("Enter the elements of 1st matrix\n");

/\* Reading two dimensional Array with the help of two for loop. If there was an array of 'n' dimension, 'n' the numbers of loops are needed for inserting data to array.\*/

for(i=0;i<2;++i)

for(j=0;j<2;++j){

printf("Enter a%d%d: ",i+1,j+1);

scanf("%f",&a[i][j]);

}

printf("Enter the elements of 2nd matrix\n");

for(i=0;i<2;++i)

for(j=0;j<2;++j){

printf("Enter b%d%d: ",i+1,j+1);

scanf("%f",&b[i][j]);

}

for(i=0;i<2;++i)

for(j=0;j<2;++j){/\* Writing the elements of multidimensional array using loop. \*/

c[i][j]=a[i][j]+b[i][j]; /\* Sum of corresponding elements of two arrays. \*/

}

printf("\nSum Of Matrix:");

for(i=0;i<2;++i)

for(j=0;j<2;++j)

{

printf("%.1f\t",c[i][j]);

if(j==1)

/\* To display matrix sum in order. \*/

printf("\n");

}return 0;

}

**Ouput**

Enter the elements of 1st matrix

Enter a11: 2;

Enter a12: 0.5;

Enter a21: -1.1;

Enter a22: 2;

Enter the elements of 2nd matrix

Enter b11: 0.2;

Enter b12: 0;

Enter b21: 0.23;

Enter b22: 23;

Sum Of Matrix:

2.2 0.5

-0.9 25.0