**DATA TYPES**

Data types are used to specify what kind of data a variable can hold. There are four **basic data types** in c. They are int, char, float, double.

1. **int:**

The data type int holds the integer value and occupy 2 bytes of memory space. 2 bytes🡪 older system (16 bit system)

4 bytes🡪 modern system (32 or 64 bit system)

**Example:**

|  |  |
| --- | --- |
| Declaration: | Initialization |
| int a,b; | int a=5;  int b=10; |

**%d**: This format specifier is used for printing and reading integers. It represents a signed decimal integer.

* **Example (Printing an integer):**

int num = 42;

printf("The number is %d\n", num);

* **Example (Reading an integer):**

int num;

printf("Enter an integer: ");

scanf("%d", &num);

1. **char:**

The char type hold a single ASCII character within two single quotes and occupy one byte of memory space.

Example:

|  |  |
| --- | --- |
| Declaration: | Initialization |
| Char ch; | Char ch=’c’; |

**%c**: This format specifier is used for printing and reading a single character.

* **Example (Printing a character):**

char ch = 'A';

printf("The character is %c\n", ch);

* **Example (Reading a character):**

char ch;

printf("Enter a character: ");

scanf(" %c", &ch);

1. **float:**

The float type hold the value with decimal points and occupy 4 bytes of memory space.

Example:

Float a=53.254

**%f**: This format specifier is used for printing and reading floating-point numbers. It represents a floating-point number with default precision.

* **Example (Printing a float):**

float num = 3.14;

printf("The number is %f\n", num);

* **Example (Reading a float):**

float num;

printf("Enter a floating-point number: ");

scanf("%f", &num);

1. **double:**

The double data type variable occupies 8 bytes of memory space that holds the value with decimal points.

**Example:**

Double a =2.3545545

**Program:**

#include <stdio.h>

int main() {

double num1, num2, sum, product;

// Prompt the user for input

printf("Enter the first double value: ");

scanf("%lf", &num1);

printf("Enter the second double value: ");

scanf("%lf", &num2);

// Calculate the sum and product

sum = num1 + num2;

product = num1 \* num2;

// Print the results

printf("The sum of %.2lf and %.2lf is %.2lf\n", num1, num2, sum);

printf("The product of %.2lf and %.2lf is %.2lf\n", num1, num2, product);

return 0;

}

**Output:**

Enter the first double value: 53.23

Enter the second double value: 43.65

The sum of 53.23 and 43.65 is 96.88

The product of 53.23 and 43.65 is 2323.49

**USER DEFINED DATATYPE:**

**Typedef:**

typedef keyword is used to give new name to the existing data type.

### ****Syntax for Typedef in C:****

Typedef existing datatype newname;

Here,

* **typedef:** The keyword that indicates the creation of a new type.
* **Existing datatype:** The data type for which a new name is being created.
* **newname:** The new name assigned to the existing data type.

In the below program, class is the variable name (alias) given to the existing data type int. So, after this type definition, class no can be used instead of data type int.

**Program:**

#include <stdio.h>

int main()

{

typedef int it;

it a,b,c;

a=10;

b=20;

c=30;

printf("a value is %d\n",a);

printf("b value is %d\n",b);

printf("c value is %d",c);

return 0;

}

OUTPUT:

a value is 10

b value is 20

c value is 30

**ENUM:**

The enum in C is also known as the enumerated type. It is a user-defined data type that consists of integer values, and it provides meaningful names to these values. The use of enum in C makes the program easy to understand and maintain. The enum is defined by using the enum keyword.

The following is the way to define the enum in C:

**enum** flag{integer\_const1, integer\_const2,.....integter\_constN};

**For example:**

**enum** fruits{mango, apple, strawberry, papaya};

The default value of mango is 0, apple is 1, strawberry is 2, and papaya is 3. If we want to change these default values, then we can do as given below:

**enum** fruits

{

mango=2,

apple=1,

strawberry=3,

papaya=0,

};

PROGRAM:

#include <stdio.h>

// Define an enum for the days of the week

enum Day {

SUNDAY, // 0

MONDAY, // 1

TUESDAY, // 2

WEDNESDAY, // 3

THURSDAY, // 4

FRIDAY, // 5

SATURDAY // 6

};

int main()

{

int dayNumber;

// Prompt the user for a day number

printf("Enter a number (0-6) for the day of the week: ");

scanf("%d", &dayNumber);

// Use a switch statement to print the day name based on the input

switch (dayNumber)

{

case SUNDAY:

printf("The day is Sunday.\n");

break;

case MONDAY:

printf("The day is Monday.\n");

break;

case TUESDAY:

printf("The day is Tuesday.\n");

break;

case WEDNESDAY:

printf("The day is Wednesday.\n");

break;

case THURSDAY:

printf("The day is Thursday.\n");

break;

case FRIDAY:

printf("The day is Friday.\n");

break;

case SATURDAY:

printf("The day is Saturday.\n");

break;

default:

printf("Invalid number. Please enter a number between 0 and 6.\n");

break;

}

return 0;

}

Output:

Enter a number (0-6) for the day of the week: 3

The day is Wednesday.

**Derived data type**

1. **Array:**

An array is a group of related data items that share as a common name.

**Program:**

#include <stdio.h>

void main()

{

int a[2]={10,20};

printf("elements %d\n",a[0]);

printf("elements %d\n",a[1]);

}

**Output:**

elements 10

elements 20

1. **Structure:**

It allows you to group different types of data under a single name.

Ex: to store details about a book, it includes title, author, subject, id all within a single data structure.

Example:

Struct book

{

char title [30];

char author[20];

char subject [ 20];

int book\_id;

};

Each stores different bytes i.e. 30+20+20+2 =72 bytes.

1. **Union:**

A union is similar to structure, but the key difference is how it stores the data.

union Book

{

char title[30];

char author[20];

char subject[20];

int book\_id;

};

Only one member of a union can hold a value at any given time.

**Note:**

**Structure:**

Allocate memory space separately for all its members.

**Union:**

Single shared memory for all its members. It takes higher memory space in the given list.

1. **Pointer:**

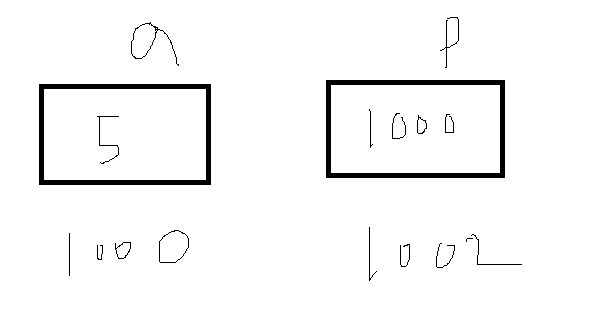
A **pointer** is a variable that stores the **memory address** of another variable. Instead of holding a data value directly, a pointer holds the location where the data is stored.

**program:**

int a=5;

int \*p;

p=&a;



Here “a” is name, “5” is value and “1000” is memory address.