20 POINTERS

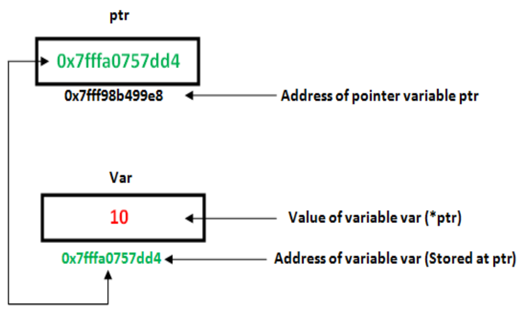
POINTERS: A **pointer** however, is a variable that stores the memory address as its value**.**

- A pointer variable points to a data type (like int or string) of the same type and is created with the * operator.

SYNTAX: datatype *var_name;

```
int *ptr;
```

- **HOW TO USE POINTER? :**
  - Define a pointer variable
  - Assigning the address of a variable to a pointer using unary operator (&) which returns the address of that variable.
  - Accessing the value stored in the address using unary operator (*) which returns the value of the variable located at the address specified by its operand.


ptr
0x7fffa0757dd4
0x7fff98b499e8
Address of pointer variable ptr
Var
10
Value of variable var (*ptr)
0x7fffa0757dd4
Address of variable var (Stored at ptr)

PROGRAM:

```
#include <iostream.h>

void pointer()
{
   int var = 20;

   // declare pointer variable
   int *ptr;

   // note that data type of ptr and var must be same
   ptr = &var;

   // assign the address of a variable to a pointer
Cout<<"Value at ptr"<<ptr<<endl;
Cout<<"Value at var"<<var<<endl;
Cout<<"Value at *ptr"<<*ptr<<endl;

int main()
{
   pointer();
}
```

OUTPUT:

```
Value at ptr = 0x7ffcb9e9ea4c
Value at var = 20
Value at *ptr = 20
```

- **POINTER TO ARRAY:**
  - An array name acts like a pointer constant.
  - The value of this pointer constant is the address of the first element.
  - For example, if we have an array named val then **val** and **&val[0]** can be used interchangeably.

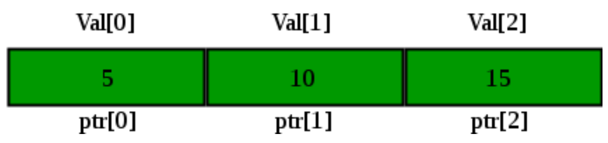
PROGRAM:

```
 using namespace std;


void geeks()
{
   // Declare an array
   int val[3] = { 5, 10, 15};
```

```
    // Declare pointer variable
    int *ptr;

    // Assign address of val[0] to ptr.
    // We can use ptr=&val[0];(both are same)
    ptr = val ;
    cout << "Elements of the array are: ";
    cout << ptr[0] << " " << ptr[1] << " " <<
ptr[2];

    return;
}

// Driver program

int main()
{
    geeks();
    return 0;
}
```

OUTPUT:

```
Elements of the array are: 5 10 15
```



- **POINTER TO STRING CONSTANT**:

```
# include < iostream.h >
void main ( )
{
char stu1 [ ] = "work as an array";
char *stu2 = "work as a pointer";
cout << stu1; //display work as an array
cout << stu2; // display work as a pointer
stu1 ++; // wrong statement
stu2 ++;
cout << stu2; // it prints "ork as a pointer"
}
```

- **STRUCTURE POINTER:** It is defined as the pointer which points to the address of the memory block that stores a structure is known as the structure pointer.

PROGRAM:

```
struct point
{
  int value;
};


// Driver Code
int main()
{
   struct point s;
   struct point *ptr = &s;
   return 0;
}
```

- **POINTER TO OBJECT**:

The next program creates a simple class called **My_Class**, defines an object of that class, called *ob*, and defines a pointer to an object of type **My_Class**, called p.

PROGRAM:

```
#include <iostream>
using namespace std;

class My_Class {
  int num;
public:
  void set_num(int val) {num = val;}
  void show_num();
};

void My_Class::show_num()
{
  cout << num << "\n";
}

int main()
{
  My_Class ob, *p; // declare an object and
pointer to it

  ob.set_num(1); // access ob directly
  ob.show_num();

  p = &ob; // assign p the address of ob
  p->show_num(); // access ob using pointer

  return 0;
}
```

```
// Incrementing and decrementing an object
pointer.

#include <iostream>
using namespace std;

class My_Class {
  int num;
public:
  void set_num(int val) {num = val;}
  void show_num();
};

void My_Class::show_num()
{
  cout << num << "\n";
}

int main()
{
  My_Class ob[2], *p;

  ob[0].set_num(10);  // access objects directly
  ob[1].set_num(20);

  p = &ob[0];  // obtain pointer to first element
  p->show_num(); // show value of ob[0]
using pointer

  p++;  // advance to next object
  p->show_num(); // show value of ob[1]
using pointer

  p--;  // retreat to previous object
  p->show_num(); // again show value of
ob[0]

  return 0; }
```

OUTPUT:

```
10
20
30
```

- **this POINTER:**

C++ uses a unique keyword called **this** to represent the object that invokes a member function.

This is a pointer that points to the object for which this function was called.

PROGRAM:

```
class ABC
{
int rn;
public:
void getdata ( )
{
cin >> this -> rn;
}
void putdata ( )
{ cout << this -> rn;
};
void main ( )
{
ABC  A, B;
 A . getdata ( );
A . putdata ( );
B . getdata ( );
B . putdata ( );
}
```

When a getdata ( ) or putdata ( ) function is called through object A, this has the address of object A. Similarly, when a getdata ( ) or putdata ( ) function is called through object B, **this** has the address of object B.

CHECK YOURSELF

1. Which of the following is the correct way to declare a pointer ?

   A. int *ptr
   B. int ptr
   C. int &ptr
   D. All of the above
2. A pointer can be initialized with

   A. Null
   B. Zero
   C. Address of an object of same type
   D. All of the above

3. The operator used for dereferencing or indirection
   is _____
   A) *
   B) &
   C) ->
   D) –>>
4. Choose the right option:
   String *x,y;
   A) X is a pointer to a string, y is a string
   B) Y is a pointer to a string, x is a string
   C) Both x and y are pointers to string types
   D) Y is a pointer to a string

5. Referencing a value through a pointer is called

   A. Direct calling
   B. Indirection
   C. Pointer referencing
   D. All of the above

STRETCH YOURSELF

1. What is a pointer and give a suitable example describing use of it?
2. Give an example of array of pointers.
3. Define this pointer. Give an example of this pointer.

ANSWERS

Answers to Check Yourself:
1. A
2. D
3. A
4. A
5. B