C Pointers

Pointers (pointer variables) are special variables that are used to store addresses rather than values.

Pointer Syntax

Here is how we can declare pointers.

```
int* p;
```

Here, we have declared a pointer `p` of `int` type.
You can also declare pointers in these ways.

```
int *p1;
int * p2;
```

Assigning addresses to Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
```

Here, 5 is assigned to the `c` variable. And, the address of `c` is assigned to the `pc` pointer.

Get Value of Thing Pointed by Pointers

To get the value of the thing pointed by the pointers, we use the `*` operator. For example:

```
int* pc, c;
```

```
c = 5;
pc = &c;
printf("%d", *pc);   // Output: 5
```

Here, the address of `c` is assigned to the `pc` pointer. To get the value stored in that address, we used `*pc`.

Note: In the above example, `pc` is a pointer, not `*pc`. You cannot and should not do something like `*pc = &c;`

By the way, `*` is called the dereference operator (when working with pointers). It operates on a pointer and gives the value stored in that pointer.

Changing Value Pointed by Pointers

Let's take an example.

```
int* pc, c;
c = 5;
pc = &c;
c = 1;
printf("%d", c);    // Output: 1
printf("%d", *pc);  // Ouptut: 1
```

We have assigned the address of `c` to the `pc` pointer.

Then, we changed the value of `c` to 1. Since `pc` and the address of `c` is the same, `*pc` gives us 1.

Let's take another example.

```
int* pc, c;
c = 5;
pc = &c;
*pc = 1;
printf("%d", *pc);  // Ouptut: 1
printf("%d", c);    // Output: 1
```

We have assigned the address of `c` to the `pc` pointer.

Then, we changed `*pc` to 1 using `*pc = 1;`. Since `pc` and the address of `c` is the same, `c` will be equal to 1.

1. What is (void*)0?

A. Representation of NULL pointer

B. Representation of void pointer

C. Error

D. None of above

Answer: Option A

2. Can you combine the following two statements into one?

char *p;

p = (char*) malloc(100);

A. char p = *malloc(100);

B. char *p = (char) malloc(100);

C. char *p = (char*)malloc(100);

D. char *p = (char *)(malloc*)(100);

Answer: Option C

3.

In which header file is the NULL macro defined?

A. stdio.h

B. stddef.h

C. stdio.h and stddef.h

D. math.h

Answer: Option C

4.

If a variable is a pointer to a structure, then which of the following operator is used to access data members of the structure through the pointer variable?

A. .

B. &

C. *

D. ->

Answer: Option D

5.

A pointer is

A. A keyword used to create variables

B. A variable that stores address of an instruction

C. A variable that stores address of other variable

D. All of the above

Answer: Option C

6. The operator used to get value at address stored in a pointer variable is

A. *

B. &

C. &&

D. ||

Answer: Option A

7.

Point out the compile time error in the program given below.

```
#include<stdio.h>

int main()
```

```
{
   int *x;
   *x=100;
   return 0;
}
```

A. Error: invalid assignment for x

B. Error: suspicious pointer conversion

C. No error

D. None of above

Answer: Option C

8. Point out the error in the program

```
#include<stdio.h>

int main()
{
   int a[] = {10, 20, 30, 40, 50};
   int j;
   for(j=0; j<5; j++)
   {
      printf("%d\n", a);
      a++;
   }
   return 0;
}
```

A. Error: Declaration syntax

B. Error: Expression syntax

C. Error: LValue required

D. Error: Rvalue required

Answer: Option C

9. Which of the statements is correct about the program?

```
#include<stdio.h>

int main()
{
   int i=10;
   int *j=&i;
   return 0;
}
```

A. j and i are pointers to an int

B. i is a pointer to an int and stores address of j

C. j is a pointer to an int and stores address of i

D. j is a pointer to a pointer to an int and stores address of i

Answer: Option C

10.

Are the expression *ptr++ and ++*ptr are same?

A. True

B. False

Answer: Option B