G. H. Raisoni College Of Engineering And Management, Wagholi Pune

2021- 2022

Assignment no :- 8

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	10-12-2021
Subject Name /Code	Object Oriented Programming/ UTIL201/UITP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

Assignment no 8

- **Aim:** Write a program to show use of this pointers
  (1) ~~New~~ new
  (2) delete

- **Theory:→**

•• **New operation :→** 'new' operator
The new operation denotes a request for memory allocation on the free store. If sufficient memory is available, new operation initalized memory to the pointer variable.

Syntax: To use or allocate memory of any data type,
Syntax
⇒ pointer-variable = new data-type;

Here, pointer-variable is the pointer of type data-type.

Ex
```
// pointer initialized with NULL
// Then request memory for the variable
int *p = NULL;
    p = new int;
        or
// combine declaration of pointer and their assignment
int *p = new int;
```

•• **Initialize memory :** We can also initialize the memory using new operator:
⇒ pointer-variable = new data-type (value);

Ex

```
int *p = new int(25);
float *q = new float(75.25);
```

• Allocate block of memory :→

new operator is also used to allocate a block (an array) of memory of type data-type.

```
pointer-variable = new data-type [size];
```

where,

size (a variable) specifies the number of elements in an array.

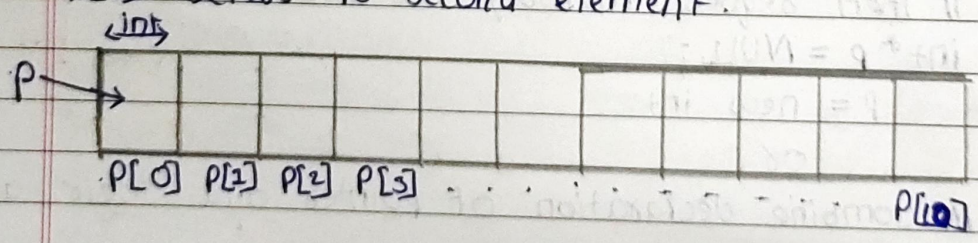
Ex :→ `int *p = new int[10]`

↓

Dynamically allocates memory for 10 integers continuously of type int and returns pointer to the first element of the sequence, which is assigned to p (a pointer).

p[0] refers to first element

p[1] refers to second element.



• Normal array declaration vs Using new

There is a difference between declaring a normal array and allocating a block of memory using new. The most important difference is, normal arrays are

deallocated by compiler (if array is local, then deallocated when function returns or completes). However, dynamically allocated arrays always remains there until either they are deallocated by programmer or program terminates.

What if enough memory is not available during runtime?

If enough memory is not available in the heap to allocate, the new request indicates failure by ~~throwi~~ throwing an exception of type std::bad_alloc, ie
unless "nothrow" is used with the new operator, in which case it returns a NULL pointer
∴ it may be good idea to check for the pointer variable produced by new before using it program.

```
int* p = new (nothrow) int;
if(!p)
{
cout<<" Memory allocation failed \n";
}
```

•• <u>delete operator</u>

Since it is programmer's responsiblity to deallocate dynamically allocated memory, programmer's are provided delete operator in C++.

Syntax:

```
// Relese memory pointed by pointer-variable
delete pointer-variable;
```

Here, pointer-variable is the pointer that points to the data object created by new.

Ex

```
delete p;
delete q;
```

To free the dynamically allocated array pointed by pointer-variable, use following form of delete

```
// Release block of memory
// Pointed by pointer-variable
delete [] pointer-variable;
```

Program code

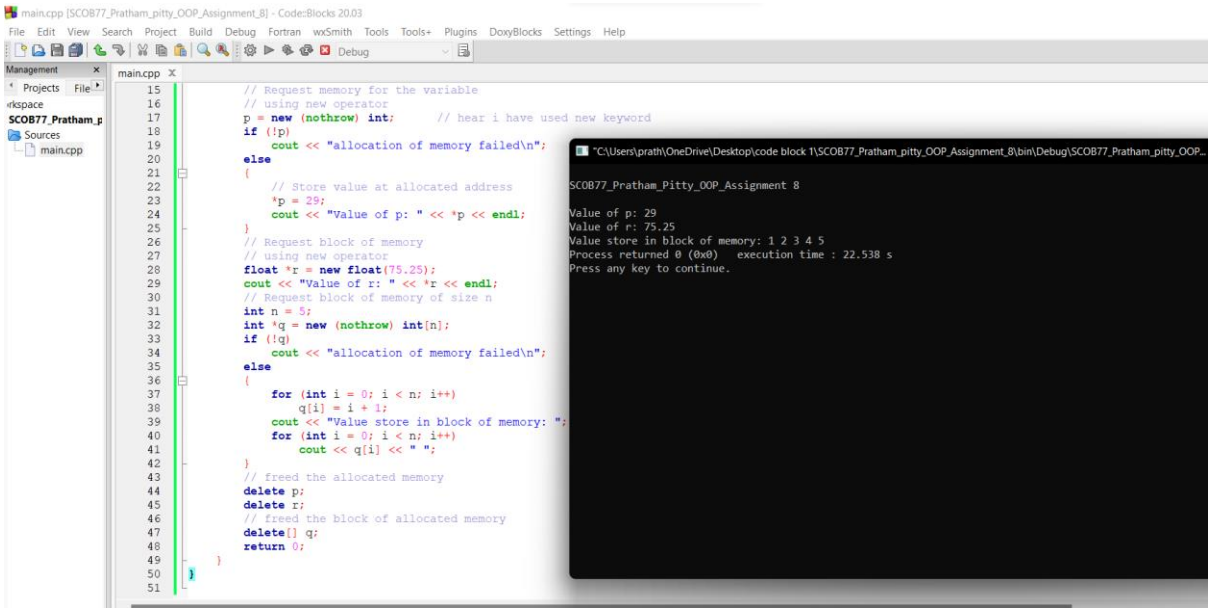
```
// Aim : Write a program to show use of this pointer, new and delete.


// C++ program to illustrate dynamic allocation
// and deallocation of memory using new and delete
#include <iostream>
using namespace std;
int main()
{
   {
      cout << "\nSCOB77_Pratham_Pitty_OOP_Assignment 8\n"
          << endl;


      // Pointer initialization to null
      int *p = NULL;
      // Request memory for the variable
      // using new operator
      p = new (nothrow) int;      // hear i have used new keyword
      if (!p)
         cout << "allocation of memory failed\n";
      else
      {
         // Store value at allocated address
         *p = 29;
         cout << "Value of p: " << *p << endl;
      }
      // Request block of memory
      // using new operator
      float *r = new float(75.25);
      cout << "Value of r: " << *r << endl;
```

```
    // Request block of memory of size n
    int n = 5;
    int *q = new (nothrow) int[n];
    if (!q)
      cout << "allocation of memory failed\n";
    else
    {
      for (int i = 0; i < n; i++)
        q[i] = i + 1;
      cout << "Value store in block of memory: ";
      for (int i = 0; i < n; i++)
        cout << q[i] << " ";
    }
    // freed the allocated memory
    delete p;
    delete r;
    // freed the block of allocated memory
    delete[] q;
    return 0;
  }
}
```

Output :-


main.cpp [SCOB77_Pratham_pitty_OOP_Assignment_8] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects File
rkspace
SCOB77_Pratham_p
Sources
main.cpp
main.cpp
15 // Request memory for the variable
16 // using new operator
17 p = new (nothrow) int; // hear i have used new keyword
18 if (!p)
19 cout << "allocation of memory failed\n";
20 else
21 {
22 // Store value at allocated address
23 *p = 29;
24 cout << "Value of p: " << *p << endl;
25 }
26 // Request block of memory
27 // using new operator
28 float *r = new float(75.25);
29 cout << "Value of r: " << *r << endl;
30 // Request block of memory of size n
31 int n = 5;
32 int *q = new (nothrow) int[n];
33 if (!q)
34 cout << "allocation of memory failed\n";
35 else
36 {
37 for (int i = 0; i < n; i++)
38 q[i] = i + 1;
39 cout << "Value store in block of memory: ";
40 for (int i = 0; i < n; i++)
41 cout << q[i] << " ";
42 }
43 // freed the allocated memory
44 delete p;
45 delete r;
46 // freed the block of allocated memory
47 delete[] q;
48 return 0;
49 }
50 }
51
"C:\Users\prath\OneDrive\Desktop\code block 1\SCOB77_Pratham_pitty_OOP_Assignment_8\bin\Debug\SCOB77_Pratham_pitty_OOP...
SCOB77_Pratham_Pitty_OOP_Assignment 8
Value of p: 29
Value of r: 75.25
Value store in block of memory: 1 2 3 4 5
Process returned 0 (0x0) execution time : 22.538 s
Press any key to continue.