C++ Dynamic Memory

Dynamic Variables are created at runtime in the memory heap.

These are ***nameless*** variables, and can be only accessed through ***pointers***.

`new` and **`delete`** operators are used to create and to destroy these dynamic variables.

`new` **operator** allocates nameless memory for a dynamic variable and ***returns a pointer*** to it.

`new` can be used to create both dynamic variables and ***dynamic arrays***:

```
new dataType;//allocates a dynamic
             //variable
```

```
new dataType[intValueOrVariable];
  //allocates an array of variables
```

`new` operator

```
int *p = new int; //Creates a
      //nameless variable at
      //runtime on the memory heap
      //and stores its address in p
```

Must ***dereference*** the pointer to access variable:

```
*p = 15;
int x = *p; //dynamic variables
  //cannot be accessed directly
  //by name: they are nameless!
```

`delete` **operator** is needed to avoid **memory leaks**: previously allocated memory that cannot be reallocated. Must ***destroy*** no longer needed dynamic variables to free it up.

```
int *p = new int(5);//initializes
                    //*p to 5

delete p;   //deallocates *p
```

The storage allocated to `*p` is reclaimed.

`delete` operator

...however, the pointer variable `p` ***still exists*** and points to the place in memory that once stored **`5`**.

`p` is now a ***dangling pointer***, which will cause problems later if dereferenced.

To avoid dangling pointer, set `p` `to` `nullptr`:

```
p = nullptr; //no dangling pointer
```

Pointer Assignment, Comparison

Value of one pointer variable can be assigned to another pointer of same type.

Two pointer variables of same type can be compared for equality: **==** , **!=**

However results of relational operators,
> , **>=** , **<** , **<=** are ***undefined***.

Pointer Arithmetic is dangerous. It can change values of undefined memory locations ***without warning***.

Address stored in one pointer can be subtracted from that of another to find the ***offset***.

Integer values can be added or subtracted from a pointer variable, ***but not*** multiplied or divided.

Results are different from integer arithmetic: they depend on **`sizeof(variableType)`**.

Dynamic Arrays are created during program execution:

```
int *p;//creates an int pointer

p = new int[6]; //allocates memory
  //for an array of 6 ints and sets
  //p to starting element's address
```

```
*p = 18;  //sets zero element to 18

p++; //points to next array element

*p = 44;  //sets next element to 44
```

Dynamic Array elements can also be accessed with array notation:

```
p[0] = 18; //these commands
p[1] = 44; //have the same result
```