ITEC243 – Lecture Session – Dynamic Memory Allocation

C Language: malloc() / malloc free(), calloc()

C++ Language: OPERATORS → **new** (memory allocation)**, delete** (memory deallocation)

- **new/delete** invokes constructor/destructor. Malloc/free will not.
- **new** does not need typecasting. Malloc requires typecasting that retruns the pointer.
- **new/delete** operators can be overloaded, malloc/free cannot.
- **new** does not require you to explicitely calculate the quantity of the memory required. (unlike malloc.)

Rather than allocating **FIXED** size to handle large size of storage, we may dynamically allocate the memory using (**new**) operator.

We will be creating **dynamic** objects (**new**) in order to allocate memory dynamically. You don't have to wait until end of the block/code to destroy the dynamic object. You have right to destroy dynamic objects any part of your block/code.

WARNING: If you forget using **delete** operator for dynamically created objects, those objects will reside in the memory.

How to create a dynamic object- Syntax: (we have to work along with pointers to create dynamic objects!!!!!)

Class_name *ptr;

1. ptr=**new** class_name; //to invoke the default constructor
2. ptr=**new** class_name(list of parameters); //to invoke the parameterized constructor
3. ptr=**new** class_name[size]; //to create a dynamic array object that invokes the default constructor

Then you should not forget to use **delete** operator to destroy the dynamic objects.

Example:

```
//time.h
class time{
private:
	int hour, minute;
public:
/*	time()
	{
		this->minute = 0;
		this->hour = 0;
	}
	time(int hour, int minute)
	{
		this->minute = minute;
		this->hour = hour;
	}*/
//Instead of creating two constructors (default and parameterized) we can do it in the following way
	time(int hour=0, int minute=0)
	{
		this->minute = minute;
		this->hour = hour;
	}
	~time() //user-defined destructor
	{
		cout << "The time object with hour and minute values:" << this->hour << ":"
			<< this->minute << " has been destroyed." << endl;
	}
	void settime(int hour=0, int minute=0)
	{
		this->hour = hour;
		this->minute = minute;
	}
	void printTime()
	{
		cout << this->hour << ":" << this->minute << endl;
	}
};
```

First main() function

```
//time.cpp
#include<iostream>
using namespace std;
#include"time.h"
void main()
{
   {
//let's create an automatic object that will invoke the default constructor
        time school;
//let's create a dynamic object that will invoke the default constructor
        time *tptr1;
        tptr1 = new time;
        (*tptr1).printTime();
        //tptr1->printtime();
//let's create another dynamic object that will invoke the parameterized constructor
        time *tptr2 = new time(8, 30);

        delete tptr1; //first dynamic object is deleted here!!!

        tptr2->settime(12, 45);
        tptr2->printTime();

        delete tptr2; //second dynamicv object is deleted here!!!

        school.settime(9, 0);
        school.printTime();
  }
        system("pause");
}
```

//Second main() function

```
#include<iostream>
using namespace std;
#include"time.h"
void main()
{
//let's create a dynamic array object for time class that the default constructor is invoked
        time *arrptr = new time[5];
        for (int i = 0; i < 5; i++)
                arrptr[i].printTime();
                delete[]arrptr;
        system("pause");
}
```