* To understand placement new and placement delete, let’s create a class Student as shown below…

|  |
| --- |
| class Student {  char\* \_name = nullptr;  *size\_t* \_size = 0;  int \_age = 0;  int \_ID = 0;  public:  Student() = default;  Student(const char\* name, int age, int id) :  \_age(age), \_ID(id) {  \_size = *strlen*(name) + 1;  \_name = new char[\_size];  *strcpy*(\_name, name);  }  ~Student() {  *cout* << "Deleting the memory!!!\n";  delete[] \_name;  \_name = nullptr;  }  // For printing purpose!!!  friend *std*::*ostream*& operator<< (*std*::*ostream*& os,  const Student& other) {  os << "Name: " << other.\_name << *endl* <<  "Age: " << other.\_age << *endl* <<  "ID: " << other.\_ID << *endl*;  return os;  }  }; |

* In the main() function, let’s define an array of 10 Students on stack.

int main(void) {

*cout* << "\n------------------------------------------------\n";

Student myStudents[10]; // Allocated memory for 10 students on stack.

* Since, allocating on stack just calls the default constructor, it looks something like this…

|  |
| --- |
| ┌───────────────────────┐  │ │  │ myStudent[0] │ ┌──────────────┐  │ │ │ │  ├───────────────────────┤ │ char\* \_name = nullptr;  │ │ │ size\_t \_size = 0;  │ myStudent[1] ├────────>┤ int \_age = 0;  │ │ │ int \_ID = 0;  ├───────────────────────┤ │ │  │ │ └──────────────┘  │ myStudent[2] │  │ │  ├───────────────────────┤  │ │  │ │  │ │  │ . │  │ . │  │ . │  │ │  │ │  │ │  ├───────────────────────┤  │ │  │ myStudent[9] │  │ │  └───────────────────────┘ |

* Placement new is a variation new operator in C++.
* Normal new operator does two things:
  1. Allocates memory.
  2. Constructs an object in allocated memory.
* Placement new allows us to separate above two things.
* In placement new, we can pass a pre-allocated memory and construct an object in the passed memory.

// Here we are not allocating memory.

// We are creating an instance of the object on

// the pre-allocated memory by calling constructor!

auto ptr1 = new(&myStudents[1]) Student("Nishith", 40, 7);

auto ptr2 = new(&myStudents[2]) Student("Nadeem", 47, 9);

* Placement new calls the constructor, in the constructor, we are allocating the memory for the char pointer (\_name) and then we are copying the string into it.

|  |
| --- |
| ┌───────────────────────┐  │ │  │ myStudent[0] │ ┌──────────────┐  │ │ │ │ ┌─────────┐  ├───────────────────────┤ │ char\* \_name = ────────►│ Nishith │  │ │ │ size\_t \_size = 8; └─────────┘  │ myStudent[1] ├────────►│ int \_age = 40  │ │ │ int \_ID = 7;  ├───────────────────────┤ │ │  │ │ └──────────────┘  │ myStudent[2] │  │ │  ├───────────────────────┤  │ │  │ │  │ │  │ │  │ │  │ │  │ │  ├───────────────────────┤  │ │  │ myStudent[9] │  │ │  └───────────────────────┘ |

* Since we have constructed by calling placement new, we must also destroy after using the object.
* Destroy here means, call the destructor once done using the object.

// We are not deleting the memory of ptr1 or ptr2.

// We are just calling the destructor of the object.

ptr1->~Student();

ptr2->~Student();

Complete Program

|  |
| --- |
| class Student {  char\* \_name = nullptr;  *size\_t* \_size = 0;  int \_age = 0;  int \_ID = 0;  public:  Student() = default;  Student(const char\* name, int age, int id) :  \_age(age), \_ID(id) {  \_size = *strlen*(name) + 1;  \_name = new char[\_size];  *strcpy*(\_name, name);  }  ~Student() {  *cout* << "Deleting the memory!!!\n";  delete[] \_name;  \_name = nullptr;  }  // For printing purpose!!!  friend *std*::*ostream*& operator<< (*std*::*ostream*& os,  const Student& other) {  os << "Name: " << other.\_name << *endl* <<  "Age: " << other.\_age << *endl* <<  "ID: " << other.\_ID << *endl*;  return os;  }  };  int main(void) {  *cout* << "\n------------------------------------------------\n";  Student myStudents[10];  auto ptr1 = new(&myStudents[1]) Student("Nishith", 40, 7);  auto ptr2 = new(&myStudents[2]) Student("Nadeem", 47, 9);  *cout* << "Printing Student 1 Details using pointer: \n";  *cout* << \*ptr1;  *cout* << "Printing Student 1 Details using array: \n";  *cout* << myStudents[1];  *cout* << "Printing Student 1 Details using pointer: \n";  *cout* << \*ptr2;  *cout* << "Printing Student 1 Details using array: \n";  *cout* << myStudents[2];  ptr1->~Student();  ptr2->~Student();  *cout* << "\n------------------------------------------------\n";  } |