

Лекция 8.

Динамическая память

1. Глобальная память — снаружи всех функций, доступ к ним есть отовсюду. Разрушаются переменные только в конце выполнения программы. Чаще всего надо избегать использования глобальных переменных.
2. Автоматический вид памяти. Компилятор сам — автоматически — разрушает переменные; также автоматически выделяет память.
3. Динамическая память — объекты живут в памяти ровно столько, сколько прописано в программе.

```
for (int i = 0; i < 1000; ++i) {  
    int* ptr = new int{123};  
    delete ptr; // удаляем то, на что указывает указатель  
}
```

```
// объект, который живет в автоматической памяти;  
// указатель живет в автоматической памяти;  
// сам объект указателя живет после delete;
```

Ключевое слово (оператор) **new** создает объект и выделяет для него автоматическую память. Когда мы создали объект и потеряли к нему указатель, то мы не больше доступа к нему и не можем удалить. Называется утечкой памяти.

Вернуть из блока {...} тот указатель:

```
int* allocate() {  
    int* ptr = new int{123};  
    return ptr  
}
```

```
int main() {  
    int* ptr = allocate();  
    delete ptr;  
  
    return 0;  
}
```

Динамическая память живет не на стеке, а на “куче”, поэтому там можно хранить большие данные.

```
class UniquePtr {  
    UniquePtr(int* ptr) {  
  
    }  
}
```

```
std::unique_ptr<int> ptr_main = AllocateSmart();  
std::unique_ptr<int> ptr_main2 = ptr_main; // так нельзя
```

```
std::shared_ptr<int> s1{new int{42}};  
std::shared_ptr<int> s2;  
std::shared_ptr<int> s3;
```