Динамические одномерные массивы

Урок №25

Разбор домашней работы

Динамическая память

Теория

20 минут

Зачем нужна динамическая память?

Динамическая память

- Массивы будут иметь ровно тот размер, который необходим.
- Получим возможность создать хранилище данных по своим правилам
- Профессиональный рост - динамическая память это важный элемент программирования на C++

Стек и куча

Стек - область оперативной памяти, в которой хранятся все переменные, объявленные в программе.

Каждый раз, когда функция объявляет новую переменную, она добавляется в стек, а когда эта переменная пропадает из области видимости (например, когда функция заканчивается), она автоматически удаляется из стека. Когда стековая переменная освобождается, эта область памяти становится доступной для других стековых переменных.

Размер стека ограничен (чаще всего это 2Мб)

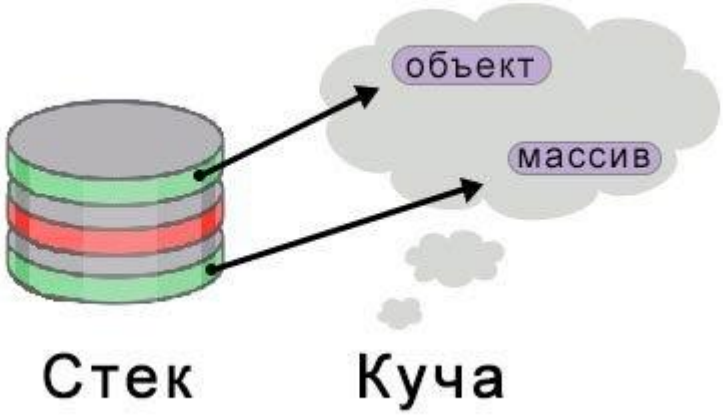
Стек и куча

Куча - вся остальная доступная для использования оперативная память.

В куче нельзя объявлять переменные, но можно выделять блоки оперативной памяти

Размер кучи ограничен только количеством оперативной памяти

Стек и куча



Динамическая память

Для выделения в куче динамической переменной используется команда new:

```
1.  int* ptr_i = new int(4); // значение инициализировано 4
2.  double* ptr_d = new double; // значение не инициализировано
```

`new` дает как результат адрес ячейки памяти, поэтому для сохранения этого адреса используется указатель

Динамическая память

После завершения работы с динамической переменной, её необходимо освободить. Для этого используется команда delete:

```
1.  int* ptr = new int(4); // значение инициализировано 4
2.  ...
3.  delete ptr;
```

Практика

7 минут

Разберем тест из классной работы

Теория

15 минут

Что из себя представляет массив в памяти?

Динамическая память

Для создания динамического массива также используется команда new, но после типа данных в квадратных скобках указывается количество элементов массива.

В отличие от обычного, для динамического массива можно как размер указать переменную:

```
1.  int N;
2.  cin >> N;
3.  int* arr = new int[N];
```

Освобождение динамического массива делается командой `delete[] arr;`

Динамический массив

Вся остальная работа с динамическим массивом происходит так же, как и с обычным массивом:

```
1.  int N;
2.  cin >> N;
3.  int* arr = new int[N];
4.  for (int i = 0; i < N; i++) {
5.      cin >> arr[i];
6.  }
7.  ...
8.  delete[] arr;
```

Преимущества динамического массива

- Динамический массив имеет строго тот размер, который необходим
- Максимальный размер динамического массива больше максимального размера обычного массива
- Динамический массив можно пересоздавать по ходу программы
- Динамический массив можно вернуть как результат функции

Пример с функцией

```
int* readArray(int N) {
  int* arr = new int[N];
  for (int i = 0; i < N; i++) {
    cin >> arr[i];
  }
  return arr;
}


...
int* arr = readArray(5);
```

Практика

25 минут

Итоги урока

1) Что такое стек и куча?
2) Как выделить фрагмент памяти в куче?
3) Как освободить фрагмент памяти, выделенный в куче?
4) Как создать динамический массив?