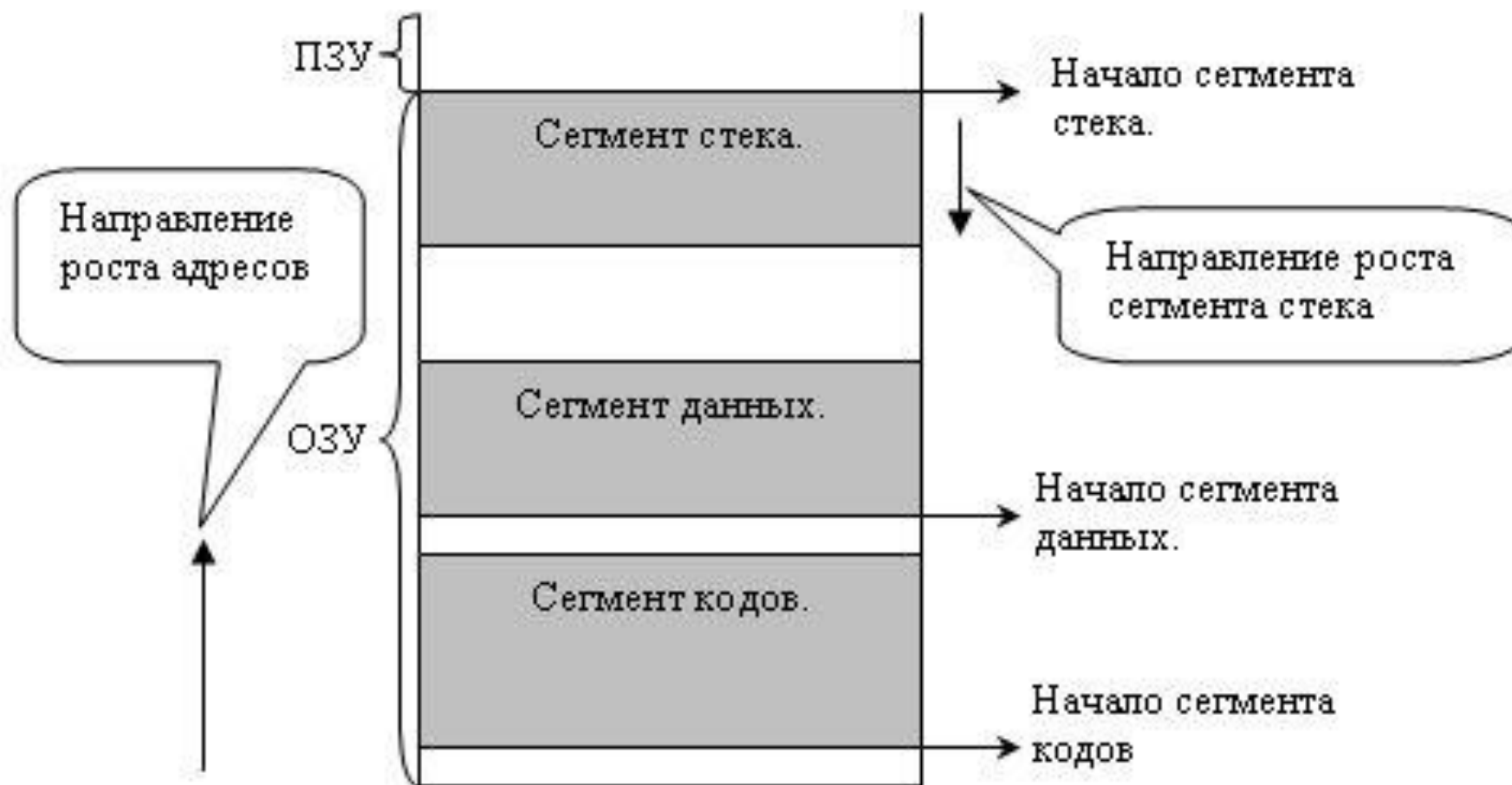


Четвертое занятие.

Управление памятью и указатели

Структура памяти программы

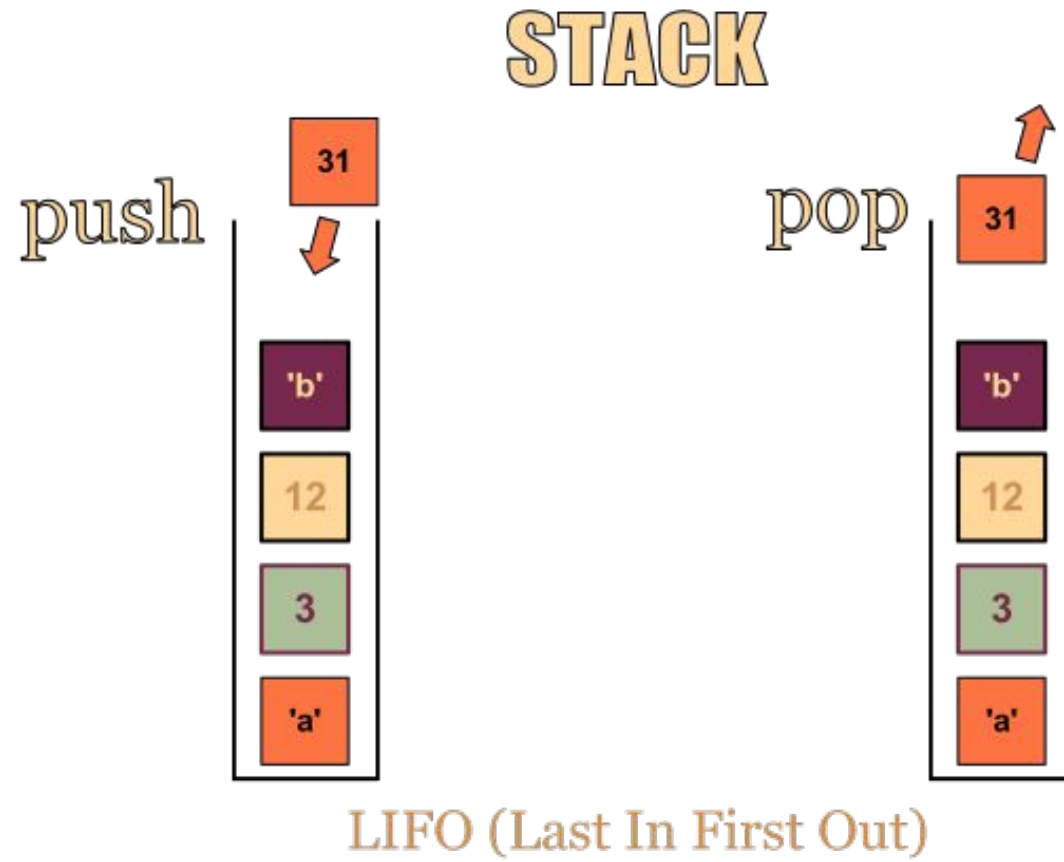


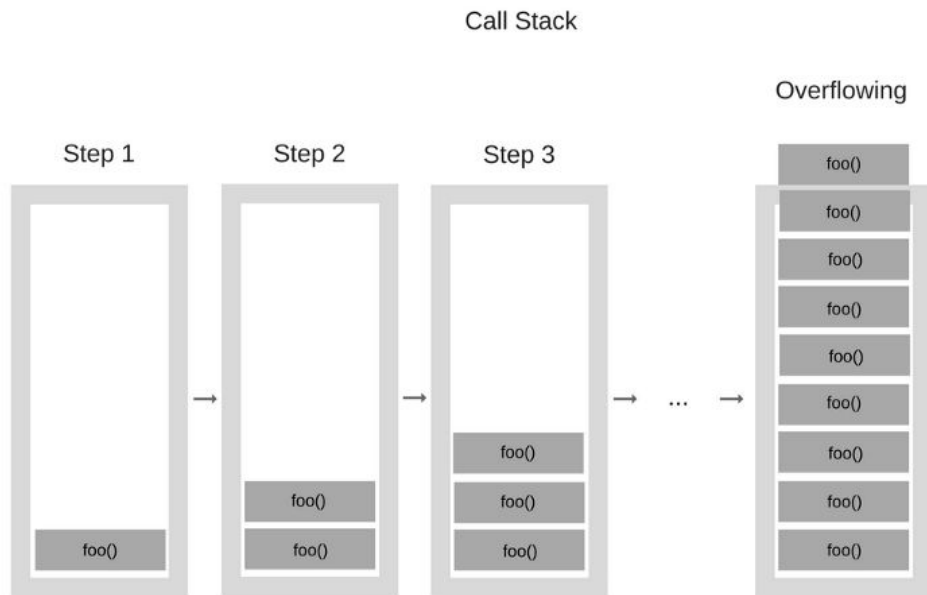
То что пока не пригодится

- **Сегменте данных** описывает переменные (выделяется память под глобальные переменные и массивы).
- **Сегмент кода** содержит команды из нашей программы, которые будут исполняться
- **Вопрос:** почему глобальные массивы и переменные создаются без мусора?

Сегмент стека (Стек)

- Стек это непрерывная область оперативной памяти организованная по принципу LIFO (last in — first out, «последним пришёл — первым вышел»).





Ограничения

- Размер данных должен быть известен до компиляции
- Размер стека ограничен (возможно переполнение стека)

Динамическая память (heap)

- структуры данных, с помощью которой реализована динамически распределяемая память приложения

Heap as an Array

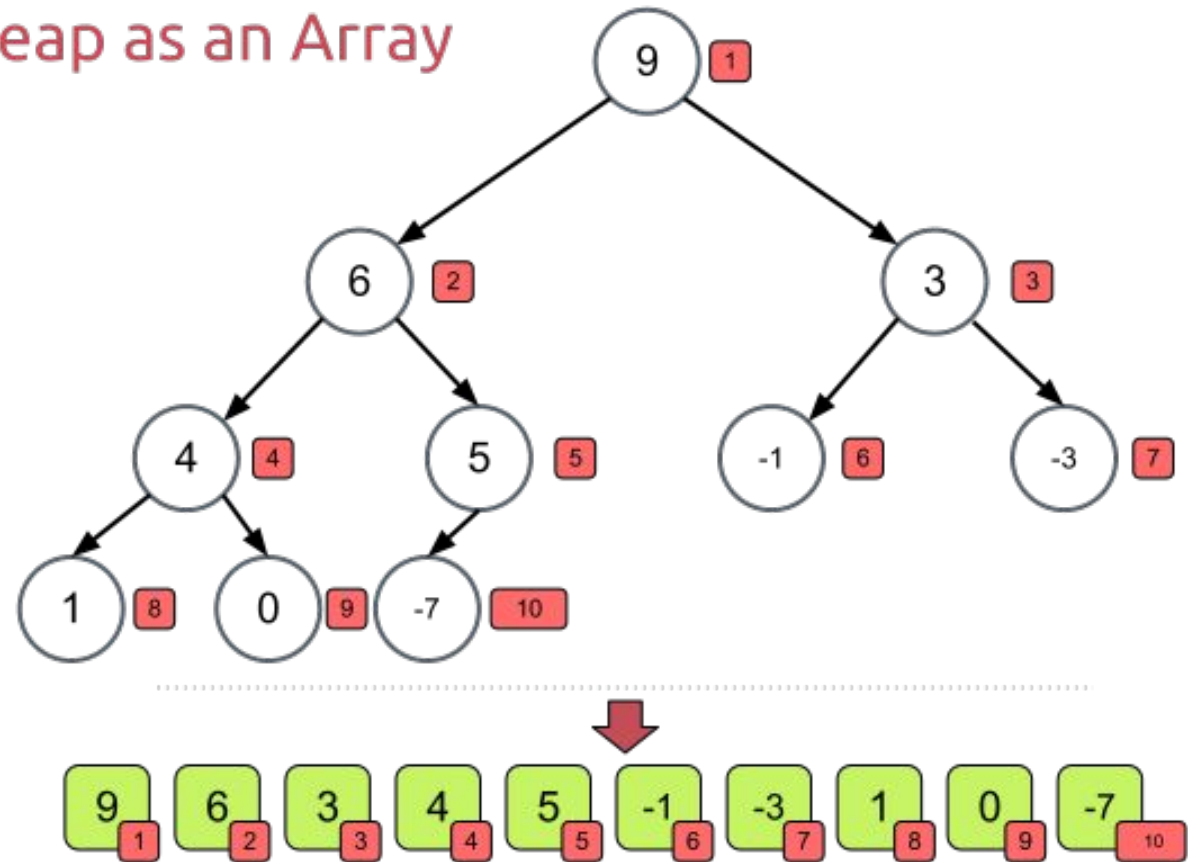
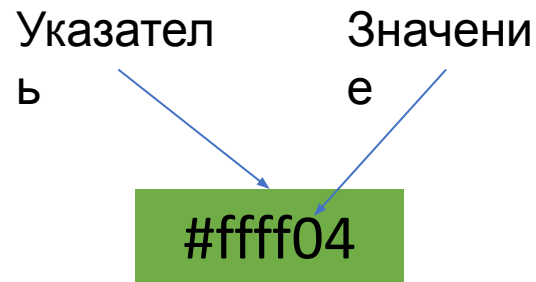


Схема выделения памяти в куче

- Создание переменной для хранения адреса (**Указатель**)
- Резервирование памяти
- Запись адреса начала памяти в **Указатель**

Указатель

- (англ. pointer) — переменная, диапазон значений которой состоит из адресов ячеек памяти или специального значения — нулевого адреса. Имеет размер разрядности приложения.



Адрес
памяти

Ячейки
памяти

Значение

#ffff01	1
#ffff02	2
#ffff03	3
#ffff04	4
#ffff05	5
#ffff06	6
#ffff07	7

The diagram shows a table with two columns. The first column contains hexadecimal addresses from #ffff01 to #ffff07. The second column contains integers from 1 to 7. Three blue arrows point to the table: one from 'Адрес памяти' (Memory address) pointing to the first column, one from 'Ячейки памяти' (Memory cells) pointing to the top row, and one from 'Значение' (Value) pointing to the second column.

Использование

- & - взятие адреса у переменной
- * - разыменование (получение значения по адресу)
- Вопрос: что выведет программа?

a = 5

b = 5

a = 3

b = 3

```
int *a;  
int b = 5;  
a = &b;  
printf("a = %d", *a);  
printf("\nb = %d", b);  
*a = 3;  
printf("\na = %d", *a);  
printf("\nb = %d", b);
```

Осторожность не помешает!

- Си позволяет легко выстрелить себе в ногу; с C++ это сделать сложнее, но, когда вы это делаете, вы отстреливаете себе ногу целиком.

Бьярне
Строуструп

Первый выстрел

```
char *a;  
int b = 2147483647;  
a = &b;  
*a = 0;
```

```
printf("%d\n", *a);  
printf("%d", b);
```

Результат
работы

0

2147483392

- $2147483647 - 2147483392 = 255$

Теперь во
вторую
ногу

```
long long number = 0;
```

```
short* array = &number;
```

```
array[0] = -184;
```

```
array[1] = -155;
```

```
array[2] = 65280 + '1';
```

```
array[3] = -148;
```

```
printf("%s", &number);
```

Зато весело

```
long long number = 0;

short* array = &number;
array[0] = -184;
array[1] = -155;
array[2] = 65280 + 'l';
array[3] = -148;

printf("%s", &number);
```

Результат с
намеком

```
C:\Course\example4\cmake-build-debug\example4.exe
H e l l
Process finished with exit code 0
```

Работа с памятью

- **malloc(N)** - выделяет блок памяти, размером N байт, и возвращает указатель на начало блока.
- **realloc(ptr, N)** - выполняет перераспределение блоков памяти. Размер блока памяти, на который ссылается параметр **ptr** изменяется на N байтов. Блок памяти может уменьшаться или увеличиваться в размере.
- **calloc(num, size)** - выделяет блок памяти для массива размером — **num** элементов, каждый из которых занимает **size** байт, и инициализирует все свои биты в нулями.

Самая важная функция

- **free(void* ptr)** - освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или realloc освобождается. То есть освобожденная память может дальше использоваться программами или ОС. **ptr** – указатель на память которую нужно освободить.

А иначе плохо

- **Утечка памяти (англ. memory leak)** - процесс неконтролируемого уменьшения объёма свободной оперативной или виртуальной памяти компьютера, связанный с ошибками в работающих программах, вовремя не освобождающих ненужные уже участки памяти



"Hey! Your application has a memory leak."

Практика

- Средне арифметическое произвольного количества чисел.

Динамический двумерный массив

- Представляет собой **указатель на указатель**.



```
int **a = (int **) calloc(n, sizeof(int));  
for (int i = 0; i < n; i++) {  
    a[i] = (int *) calloc(n, sizeof(int));  
}
```

Практика

- Организовать не прерывный ввод чисел в массив. Вывести максимальное и минимальное.

Домашнее задание

- Организовать не прерывный ввод текста в консоль. Найти самое длинное слово и вывести его.