

Семинар №13

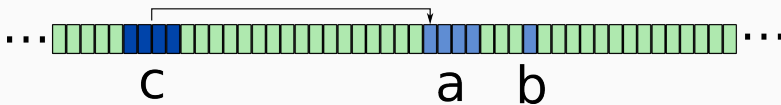
ФАКИ 2017

Бирюков В. А.

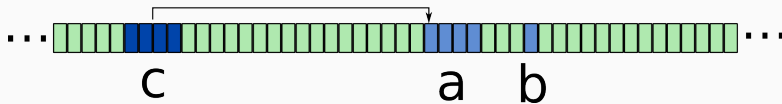
September 7, 2017

Указатели

```
int a;  
char b;  
int * c = &a;
```



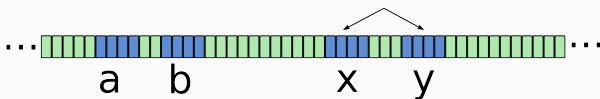
```
int a = 1;  
int * c = &a;  
*c = 5;
```



Указатели и аргументы функций

Передача по значению

```
void swap(int x, int y) /* НЕПРАВИЛЬНО! */  
{  
    int temp;  
    temp = x;  
    x = y;  
    y = temp;  
}  
...  
swap(a, b);
```



Указатели и аргументы функций

Передача по адресу

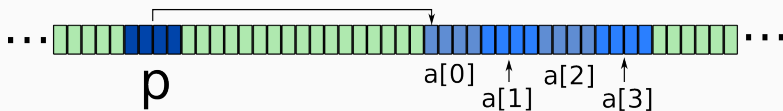
```
void swap(int * px, int * py)
{
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
...
swap(&a, &b);
```



```
int a[4] = {1, 2, 3, 4};
```

```
int * p = &a[0];
```

```
int x = *(p + 2);
```



$*(p+2)$ и $a[2]$ - это одно и то же

Malloc и free. Управление
памятью.

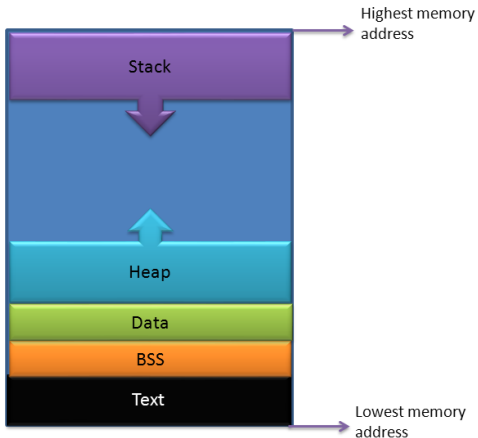


Figure : Process memory organization

- Стек представляет собой обычный алгоритмический стек, применённый для управления памяти
- В нём хранятся локальные переменные
- Имеет фиксированный размер, определяется операционной системой, на порядок меньше чем Куча
- Немного быстрее, чем Куча

- Куча представляет собой обычную алгоритмическую кучу, применённую для управления памяти
- В ней можно динамически выделять память
- Размер, обычно, ограничен только доступными ресурсами
- Немного медленней, чем Стек

Выделение памяти в Куче с помощью malloc и free

Выделение памяти на массив из 100 переменных типа int

```
int *p;
p = (int *)malloc(100 * sizeof(int));
if (p == 0) {
    printf("ERROR: Out of memory\n");
    return 1;
}

for (int i = 0; i < 100; ++i) {
    *(p+i) = 123;
}
printf("%d\n", *(p+50));

free(p);
```

Структуры

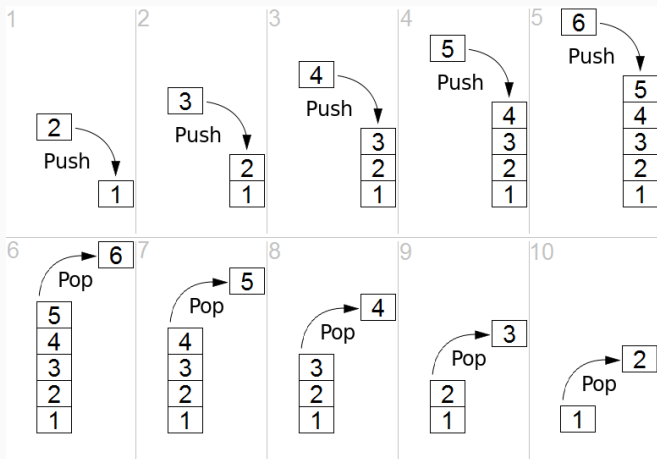
Описание структуры:

```
struct account {  
    int account_number;  
    char first_name[30];  
    char last_name[50];  
    float balance;  
};
```

Объявление структуры:

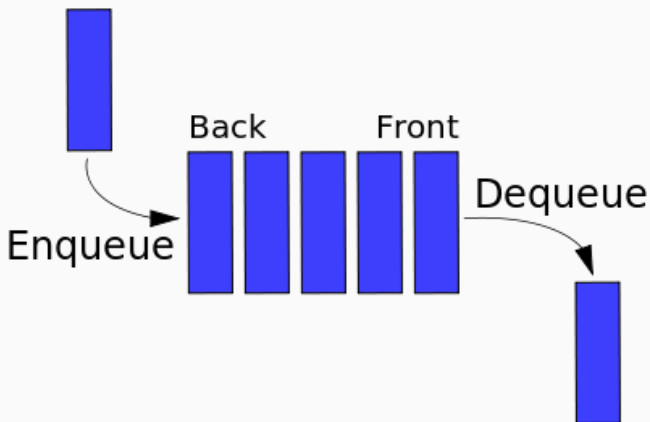
```
struct account ac1;
```

Стек и очередь




```
#define N 100
typedef int Data;

struct Stack {
    int n;
    Data a[N];
};
```



Быстрая сортировка

Быстрая сортировка (quicksort)



Неотсортированный массив



Опорное значение = 7



$12 \geq 7 \geq 2$,
меняем местами 12 и 2



$26 \geq 7 \geq 7$
меняем 26 и 7

Быстрая сортировка (quicksort)



...



- Время работы сортировки пузырьком, выбором и вставками $\sim n^2$
- Время работы сортировки слиянием и быстрой сортировки в среднем $\sim n\log(n)$

Стандартная сортировка qsort()

```
#include <stdlib.h>
int values[] = { 88, 56, 100, 2, 25 };

int cmp(const void * a, const void * b)
{
    return ( *(int*)a - *(int*)b );
}

...
qsort(values, 5, sizeof(int), cmp);
```

Функция main

```
int main ( int argc, char * argv[] )  
{  
    <Операторы>  
    return ( 0 );  
}
```

- argv – параметры, передаваемые в функцию main
- argc – количество этих параметров

```
~/ $ gcc -o prog_name prog.c
```



```
argc = 4
```

```
argv = {"gcc", "-o", "prog_name", "prog.c"}
```

Запись/чтение файлов

```
#include <stdio.h>
FILE *fptr;
fptr=fopen("output.txt", "w");
if(fptr==NULL){
    printf("Error!");
    exit(1);
}
fprintf(fptr, "%d", n);
fclose(fptr);
```

```
FILE * f = fopen("input.txt", "r");

int number_of_chars = 0;
int c;

while ((c = fgetc(f)) != EOF)
{
    number_of_chars++
}
fclose(f);
```

```
char c[] = "some string data";  
char buffer[100];  
FILE *fp = fopen("output.txt", "w+");  
fwrite(c, strlen(c) + 1, 1, fp);  
fclose(fp);
```

Структуры данных

- Структура данных (англ. data structure) — определённый способ организации данных, так, чтобы их можно было использовать эффективно.
- Для разных задач более эффективными будут разные структуры данных.

Упорядоченный массив

	Массив	Упорядоченный массив
index	$O(1)$	$O(1)$
insert	$O(1)$	$O(N)$
remove	$O(N)$	$O(N)$
find	$O(N)$	$O(\log(N))$

```
struct Node {  
    int data;  
    struct Node * next;  
};  
struct Node * head;
```

Связный список

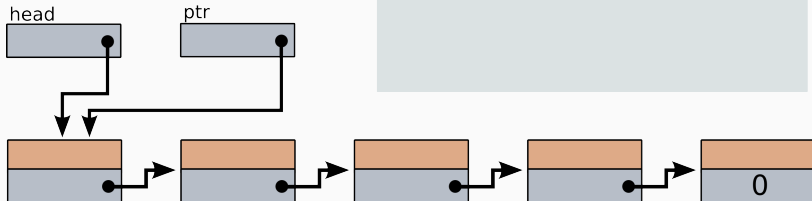
Обход связного списка - 1

Node;



Код:

```
ptr = head;
```



Связный список

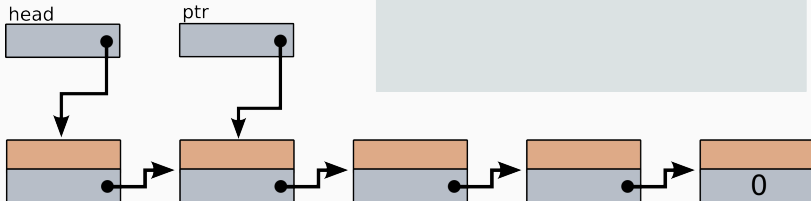
Обход связанного списка - 2

Node;



Код:

```
ptr = head;  
ptr = ptr->next;
```



Связный список

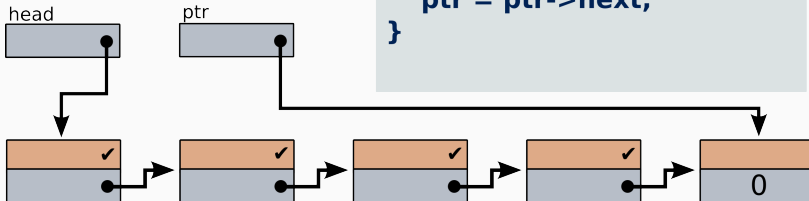
Обход связанного списка - 3

Node;



Код:

```
ptr = head;  
while ( ptr->next != 0 ) {  
    printf ("%d ", ptr->data);  
    ptr = ptr->next;  
}
```



	Список	Двусвязный список
index	$O(N)$	$O(N)$
insertToFront	$O(1)$	$O(1)$
insertToBack	$O(N)$	$O(1)$
insertAfter	$O(1)$	$O(1)$
insertBefore	$O(N)$	$O(1)$
remove	$O(1)$	$O(1)$
find	$O(N)$	$O(N)$

Задание

- Тренировочная к/р №2