

Динамические структуры данных

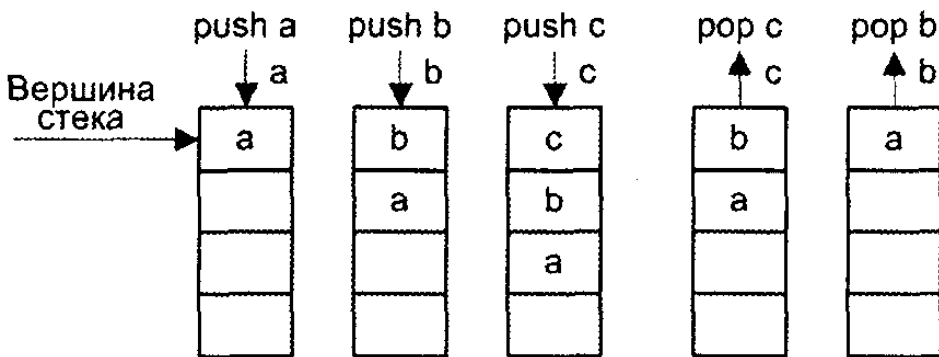
Семестр 1

Семинар 7

Простейшие структуры данных

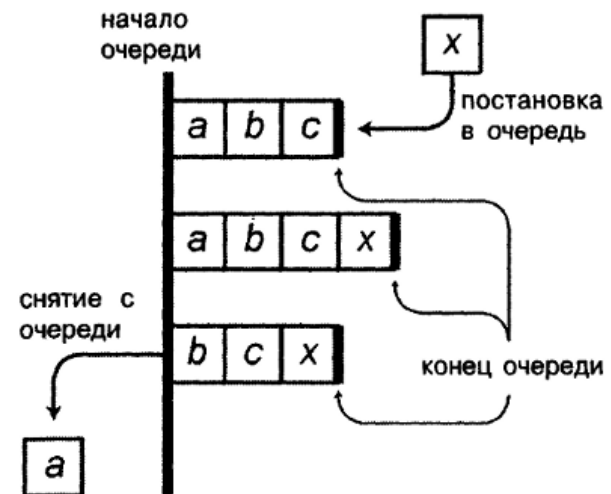
Стек

- "последним вошел — первым вышел"
- last-in, first-out — **LIFO**
- **top** — начало стека
- **push** — запись
- **pop** - извлечение



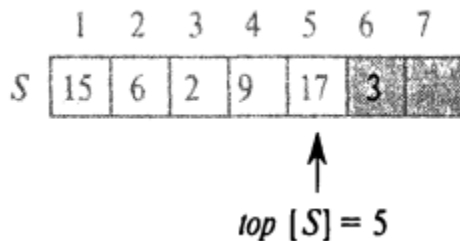
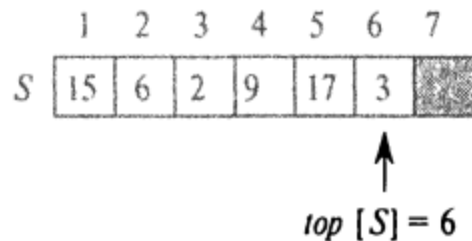
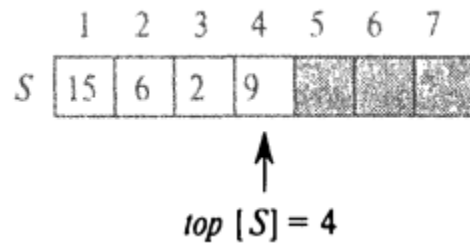
Очередь

- "первым вошел — первым вышел"
- first-in, first-out — **FIFO**
- **head** — начало очереди
- **tail** — конец очереди
- **enqueue** — запись
- **dequeue** — извлечение

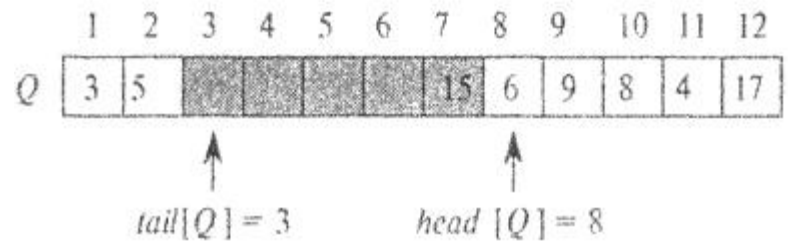
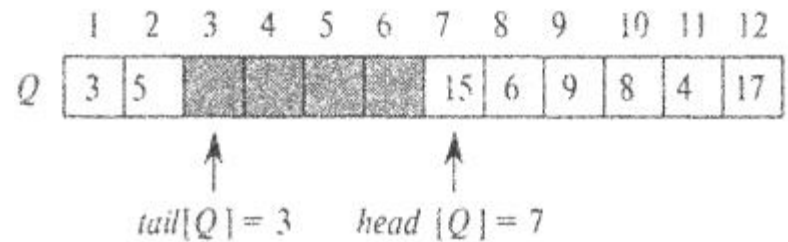
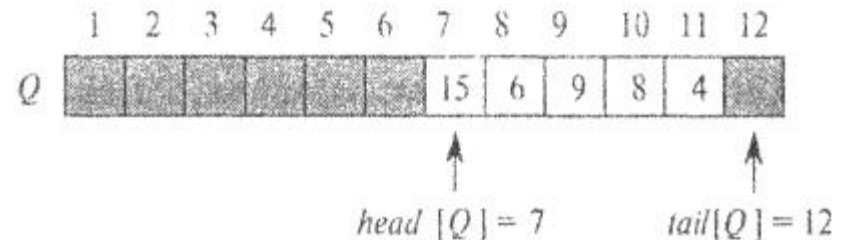


Реализация на массиве

Стек



Очередь



Реализация на массиве: код

Стек

```
struct stack {  
    int mas[N];  
    int top; };  
  
void push (struct stack *_s, int _x) {  
    if (_s->top >= N-1) return;  
    _s->top++;  
    _s->mas[_s->top] = _x; }  
  
int pop (struct stack *_s) {  
    if (_s->top < 0) return 0;  
    return _s->mas[_s->top--]; }
```

Очередь

```
struct queue {  
    int mas[N];  
    int head;  
    int tail; };  
  
void enqueue (struct queue *_q, int _x) {  
    if (_q->head - _q->tail == 1  
        || _q->head - _q->tail == N-1) return;  
    _q->mas[_q->tail] = _x;  
    if (_q->tail == N-1) _q->tail = 0;  
    else _q->tail++; }  
  
int dequeue (struct queue *_q) {  
    if (_q->head == _q->tail) return 0;  
    return _q->mas[_q->head++]; }
```

Реализация на масиве: проверка

Стек

```
int main()
{
    struct stack s1;
    s1.top = -1;
    push(&s1,10);
    push(&s1,5);
    push(&s1,7);
    printf("%d\n",pop(&s1));
    printf("%d\n",pop(&s1));
    printf("%d\n",pop(&s1));
    system("pause");
    return 0;
}
```

Очередь

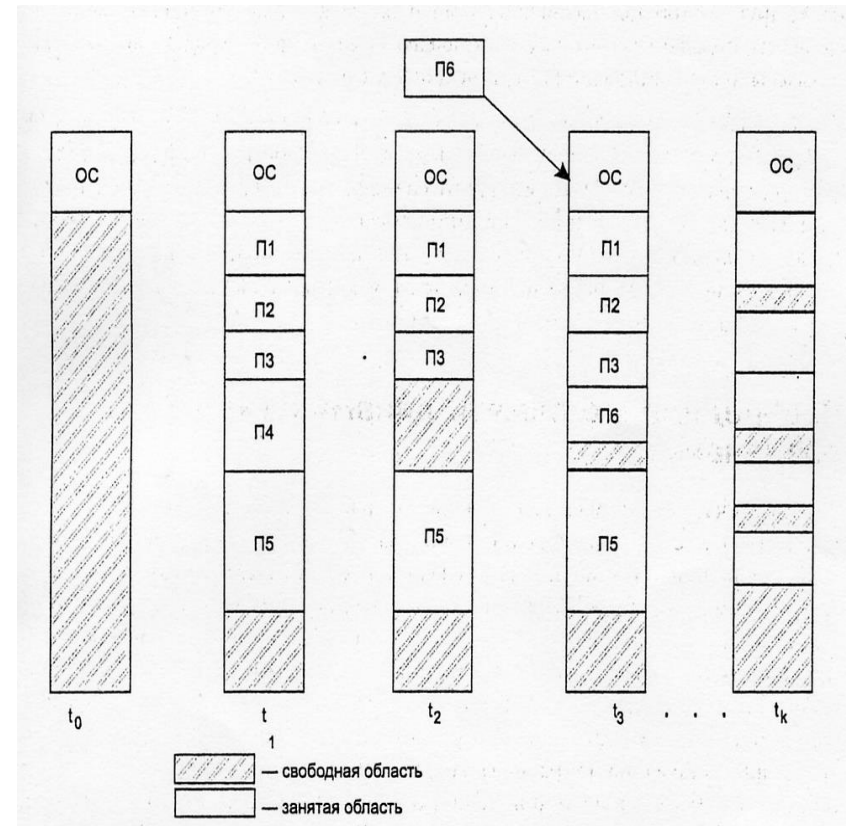
```
int main()
{
    struct queue s1;
    s1.head = s1.tail = 0;
    enqueue(&s1,10);
    enqueue(&s1,5);
    enqueue(&s1,7);
    printf("%d\n",dequeue(&s1));
    printf("%d\n",dequeue(&s1));
    printf("%d\n",dequeue(&s1));
    system("pause");
    return 0;
}
```

Задачи

0. Определить, является ли правильной введенная скобочная последовательность.
1. «Копилка». Реализовать стек, каждый элемент которого хранит три числа: номинал монеты, день и месяц. Написать функцию, которая считает общую сумму накопленного.
2. В каждой строке сначала записан номер группы (331 - 336), затем (через пробел) – фамилия студента. Необходимо вывести список студентов по группам. Использовать очереди (6 шт.).
3. Написать функцию, которая сортирует копилку по дате.

Связанные списки

- структурированные массивы
- линейный порядок
- поиск следующего элемента через предыдущий
- односвязные, двусвязные и XOR-связные
- кольцевые
- сортированные



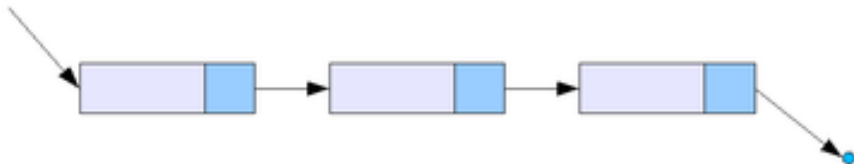
Связанные списки

Односвязные

- Singly linked list

```
struct node_single {  
    int data;  
    struct node_single * next;  
};
```

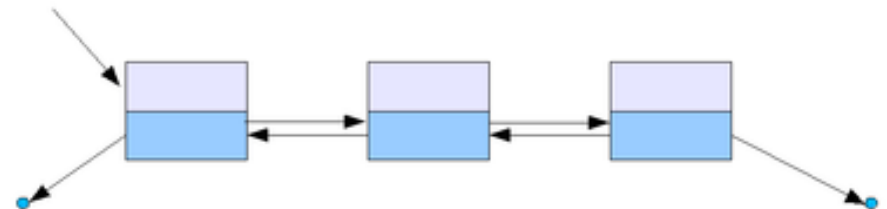
Элемент списка



Двусвязные

- Doubly linked list

```
struct node_double {  
    int data;  
    struct node_double * next;  
    struct node_double * prev;  
};
```



Связанные списки

Стек

stack_single.cpp

Очередь

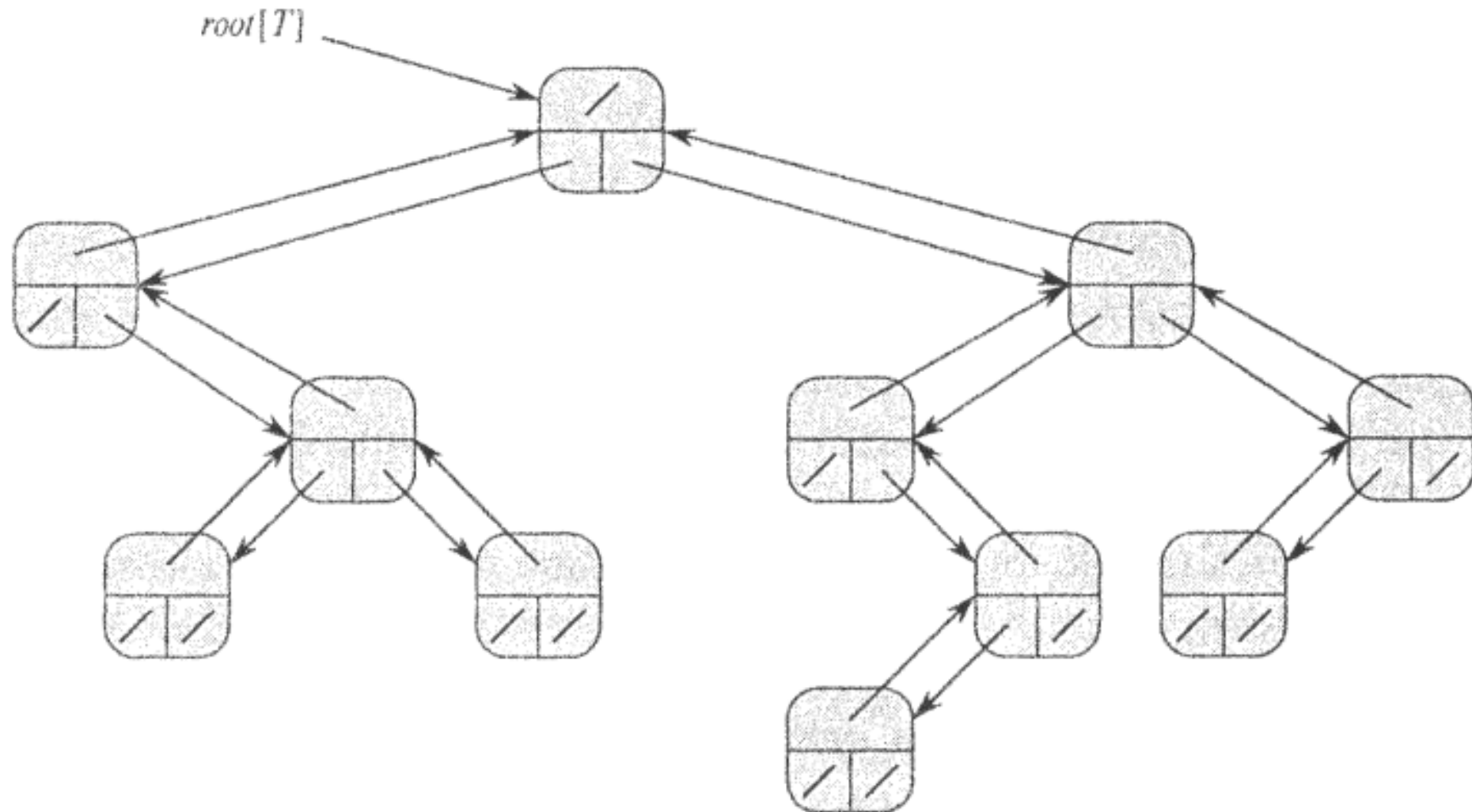
queue_single.cpp

Задача: записная книжка

С помощью односвязного списка (стека или очереди) реализовать записную книжку, каждая запись в котором содержит имя, фамилию, телефон, группу, день и месяц рождения.

0. Создать типы данных для списка и одного узла списка.
1. Функция добавления элемента в список.
2. Функции поиска человека по имени, фамилии и телефону.
3. Функция вывода всего списка на экран.
4. Функция вывода списка группы на экран.
5. Считывание данных из файла. Сохранение данных в файл.
6. Сортировка списка по дате рождения.
7. Интерфейс.

Бинарные деревья



Задачи

0. Начертите бинарное дерево, корень которого имеет индекс 6, и которое представлено приведенными ниже полями.

Индекс	<i>key</i>	<i>left</i>	<i>right</i>
1	12	7	3
2	15	8	NIL
3	4	10	NIL
4	10	5	9
5	2	NIL	NIL
6	18	1	4
7	7	NIL	NIL
8	14	6	2
9	21	NIL	NIL
10	5	NIL	NIL

Задачи

1. Реализовать функцию, которая строит дерево в соответствии с номером ключа: значение в левом потомке меньше значения в узле, значение в узле меньше значения в правом потомке.
2. Разработайте рекурсивную функцию, которая выводит ключи всех узлов бинарного дерева.
 - Обход в ширину
 - Обход в глубину
3. Реализовать функцию, которая ищет в дереве заданное значение ключа.

Подключение файлов

```
//main.cpp
```

```
#include<stdio.h>
```

```
#include<d:\Student\add.cpp>
```

```
int main()
```

```
{
```

```
    int x=55, y=9;
```

```
    printf ("%d\n", sum(x,y));
```

```
    return 0;
```

```
}
```

```
//add.cpp
```

```
int sum (int x, int y)
```

```
{
```

```
    return x+y;
```

```
}
```

Переименование типов

```
typedef struct node_single node;
typedef struct queue queue;
struct node_single {
    int data;
    node * next;
};
struct queue
{
    node * head;
    node * tail;
};
...
queue q1;
```