Динамические структуры данных

Семестр 1

Семинар 9

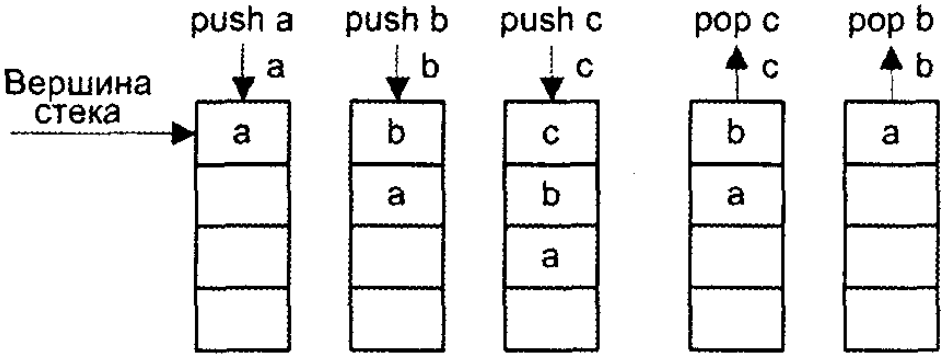
Простейшие структуры данных

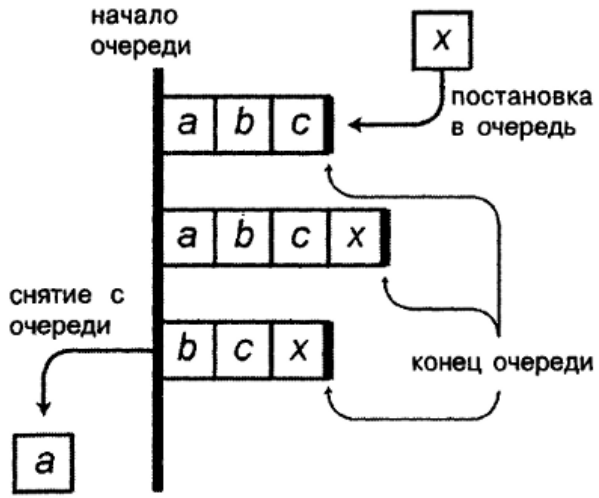
Стек

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

Очередь

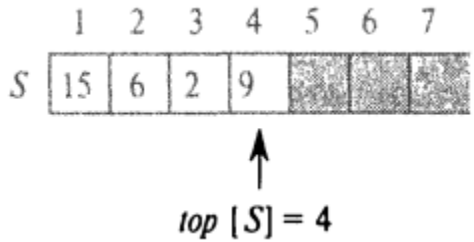
- " первым вошел — первым вышел"
- first-in, first-out — FIFO
- head – начало очереди
- tail – конец очереди
- enqueue – запись
- dequeue – извлечение

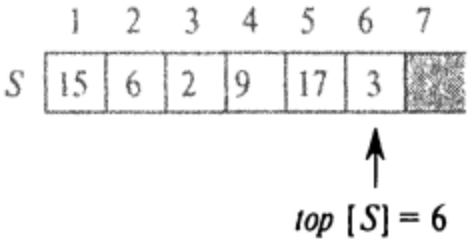


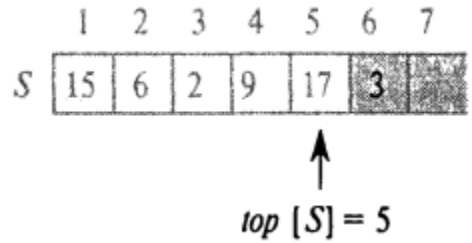


Реализация на массиве

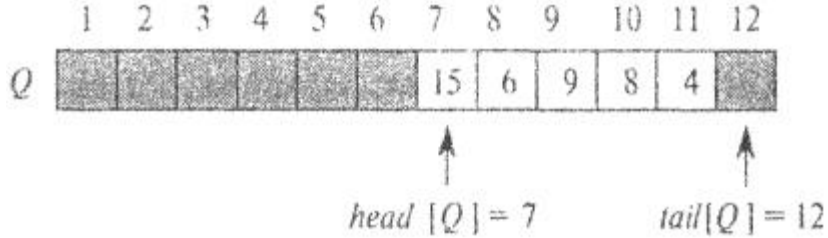
Стек

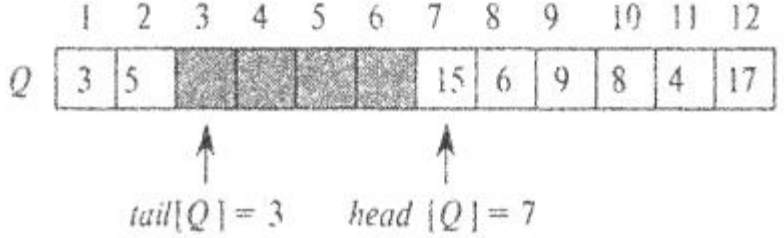


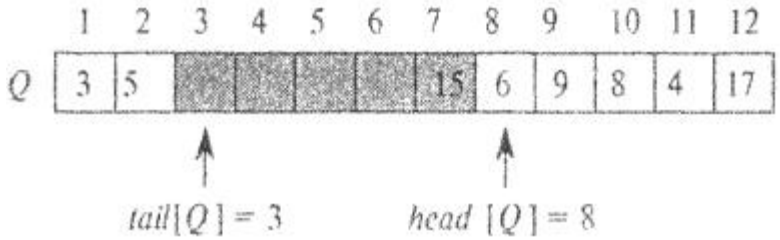




Очередь







Реализация на массиве: код

Стек

```
struct stack {
    int mas[N];
    int top; };

void push (struct stack *_s, int _x) {

int pop (struct stack *_s) {
```

Очередь

```
struct queue {
    int mas[N];
    int head;
    int tail; };
void enqueue (struct queue *_q, int _x) {

int dequeue (struct queue *_q) {
```

Реализация на массиве: 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 из символов { } [ ] ( ) < > b d.

1. «Копилка». Реализовать стек, каждый элемент которого хранит три числа: номинал монеты, день и месяц. Написать функцию, которая считает общую сумму накопленного.

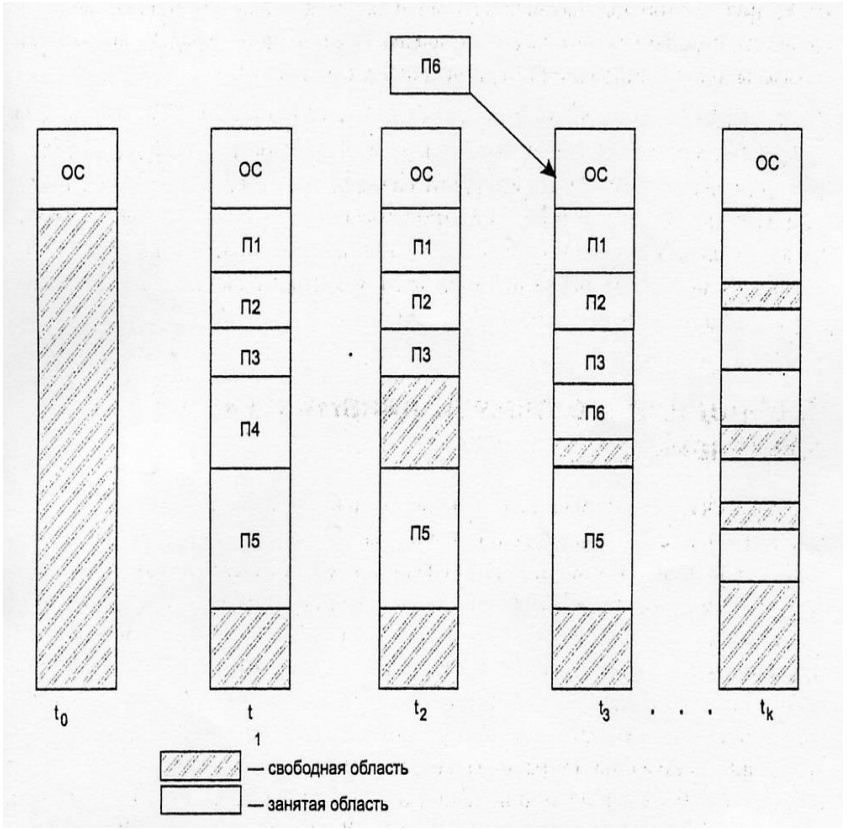
2. В каждой строке сначала записан номер группы (331 - 336), затем (через пробел) – фамилия студента. Необходимо вывести список студентов по группам. Использовать очереди (6 шт.).

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

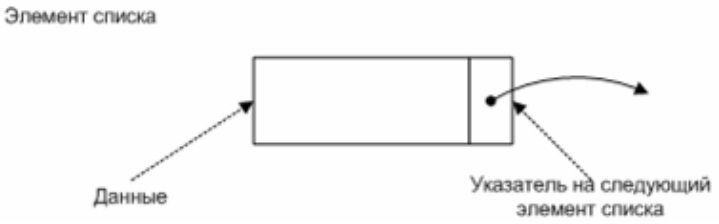


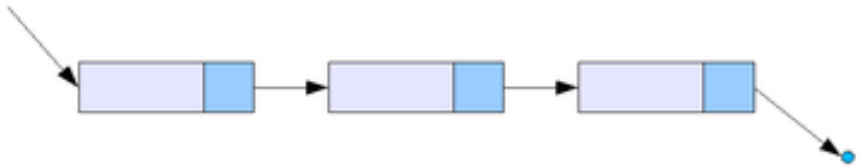
Связанные списки

Односвязные

- Singly linked list

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

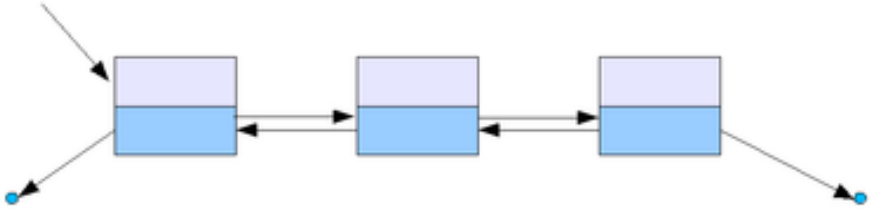




Двусвязные

- Doubly linked list

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



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 (односвязного) и узла списка.

1. Функции поиска человека по имени и фамилии.

2. Функция добавления элемента в список и удаления по имени и фамилии.

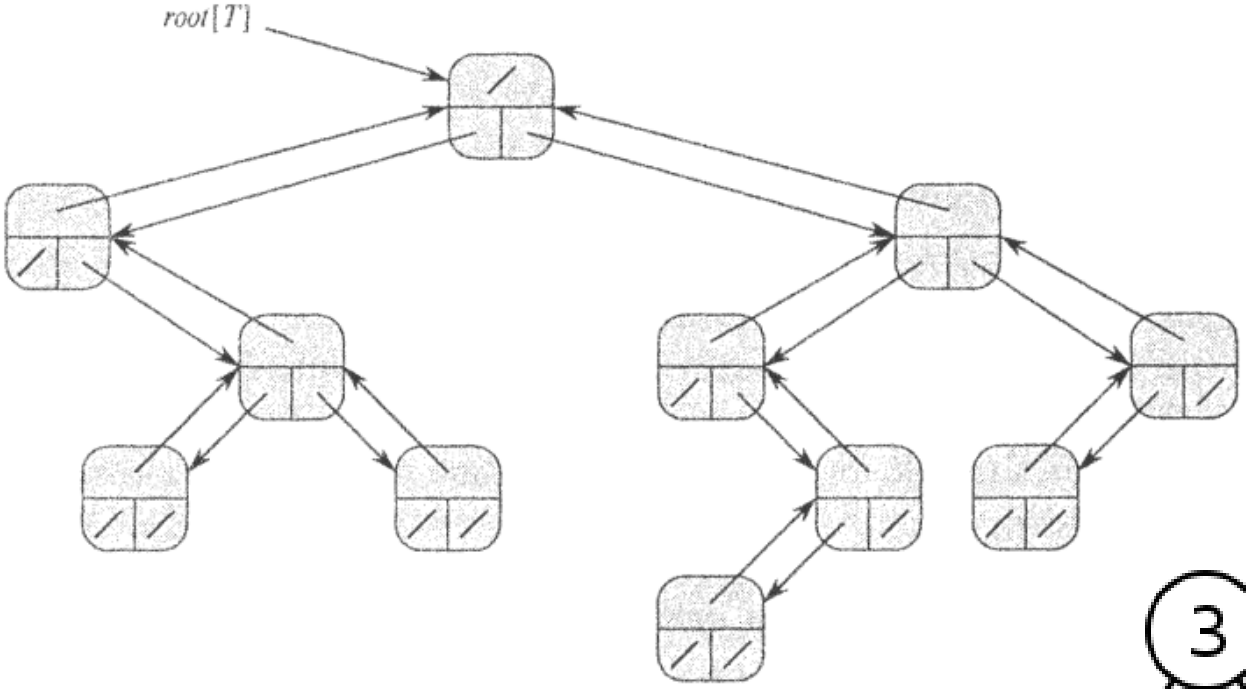
3. Функция вывода всего списка на экран.

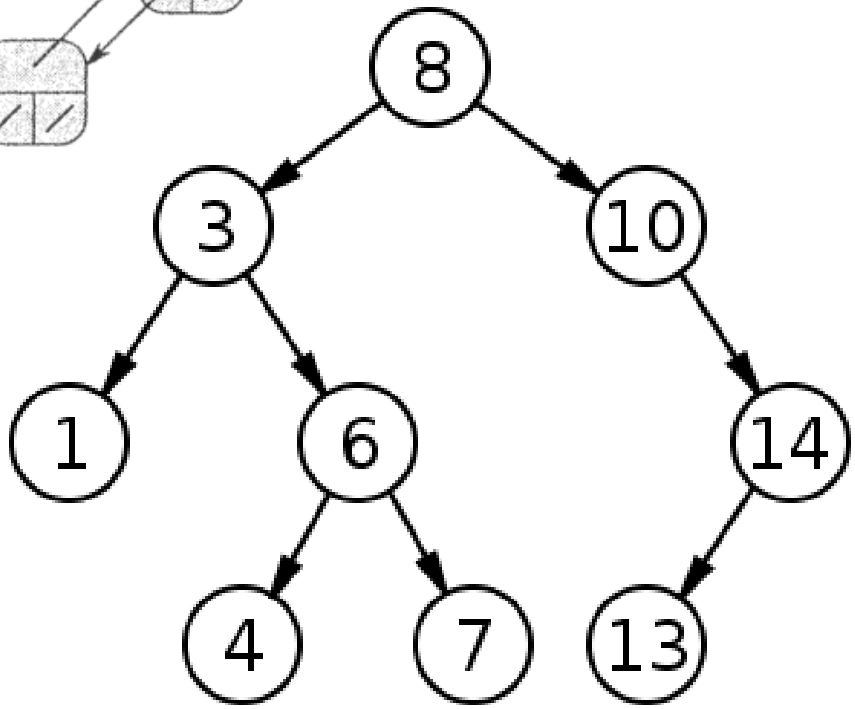
4. Функция вывода списка группы на экран.

6. Сортировка списка по дате рождения.

7. То же для двусвязного списка.

Бинарные деревья





Дерево поиска

Задачи

0. Реализуйте бинарное дерево поиска, каждый узел которого содержит, помимо ссылок на потомков, одно поле типа int и одно поле типа char*:

1) добавление элемента в дерево;

2) поиск элемента в дереве (по int);

3) удаление элемента из дерева (по int);

4) печать дерева;

5) обход дерева в глубину и ширину;

6) определение высоты дерева и поддерева;

7) одинарные повороты влево и вправо;

8) двойные повороты влево и вправо.

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;
```