

САА – Упражнение 4

Структура от данни опашка, списък

Опашката е линейна структура от данни. Операцията включване на елемент в нея е допустима само в единия край на опашката (наречен ‘край’). Операцията изключване на елемент е допустима само за другия край на опашката (наречен ‘начало’). Възможен е пряк достъп само до елемента, намиращ се в началото на опашката. При така описаната организация първият влязъл в опашката елемент се изключва първи, а последният влязъл – последен. Затова опашката се определя като структура от данни FIFO – First In First Out – ‘първи влязъл, първи излиза’.

Опашката може да бъде реализирана по два начина:

- Статично – чрез масив;
- Динамично.

Свързаният списък е крайна редица от елементи от един и същи тип. Операциите включване и изключване на елемент са допустими на произволно място в редицата. Възможен е достъп до всеки елемент от списъка, като достъпът до първия елемент от списъка е пряк, а до останалите елементи – последователен.

Примери за приложение: приоритетна опашка, отборна опашка.

В приоритетната опашка всеки елемент освен данните, които съдържа се характеризира и с едно цяло число – неговия приоритет. Когато се включва елемент в приоритетната опашка, това става не в края ѝ, а веднага след последния елемент с по-висок приоритет (ако елементите са с еднакъв приоритет наредбата е по реда на постъпването им). Така всички елементи с по-висок приоритет се намират преди него, а всички с по-нисък приоритет – след него. Изключването на елемент става от началото на опашката – там винаги се намира елементът с най-висок приоритет.

Задачи:

1. Напишете програма, която симулира приоритетна опашка, в която ще се въвеждат задачи за изпълнение. Реализирайте програмно функциите за добавяне и изтриване на елемент в приоритетната опашка, както и функция за разпечатване на опашката. Използвайте динамична реализация. Функцията за изтриване да изтрива всички елементи, които са с по-висок или равен на приоритета, зададен от потребителя. Реализирайте програмно втора функция за изтриване на определен брой елементи от опашката.

Дефинирайте структурата, която ще използвате за съхраняване на данни в приоритетната опашка и указател към началото ѝ.

```
struct queue
{
    int rank;
    char task[20];
}
```

```

        struct queue * next;
    } * head = NULL;

```

В главната функция организирате потребителския вход – въвеждането на задачите в приоритетната опашка. За всяка задача въведете име (по-кратко от 20 символа) и приоритет. Добавяйте всяка въведена задача на правилното място в приоритетната опашка. За целта функцията за добавяне на елемент трябва да съдържа проверка за определяне на правилната позиция на задачата съобразно нейния приоритет.

Във функцията за добавяне на елемент в приоритетната опашка можете да използвате следния код:

```

ptr = (struct queue *)malloc(sizeof(struct queue));
if (!ptr)
    return 0;

ptr->rank = rank;
strcpy(ptr->task,task);

// Дефинирайте функцията findPlace (), която определя мястото на новия елемент
ptr1 = findPlace (rank);

if (!ptr1)
{
    ptr->next = head;
    head = ptr;
}
else
{
    ptr->next = ptr1->next;
    ptr1->next = ptr;
}
return 1;

```

Функцията за разпечатване:

```

void printTaskQueue()
{
    struct queue * ptr = head;
    while (ptr)
    {
        printf("%s, %d\n", ptr->task, ptr->rank);
        ptr = ptr->next;
    }
}

```

2. Напишете програма, която симулира отборна опашка. Реализирайте програмно функциите за добавяне и изтриване на елемент в отборната опашка, както и функция за разпечатване на опашката. Използвайте динамична реализация.