ЛАБОРАТОРНАЯ РАБОТЫ №3
«СТЕК И ОЧЕРЕДЬ»

1.1 Цель работы

Целью работы является изучение структур данных «стек» и «очередь», а также получение практических навыков их реализации.

1.2 Задание на лабораторную работу

Реализовать структуры данных «стек» и «очередь» в соответствии с заданным вариантом. Дополнительно программа должна удовлетворять следующим требованиям:

1) Вывод на экран состояния моделируемой системы на каждой итерации работы (содержимое стека(ов), очереди(ей), процессора(ов));
2) Для каждой задачи из списка входных задач должно быть определено время поступления;
3) Необходимо наличие, как автоматического генератора задач, так и возможность ручного добавления задач, с указанием их параметров (в зависимости от задания);
4) Необходимо обработать ситуации, при которых какая-либо структура данных может быть переполнена.

Варианты задания приведены в таблице 3 (формулировки задач приведены после таблицы).

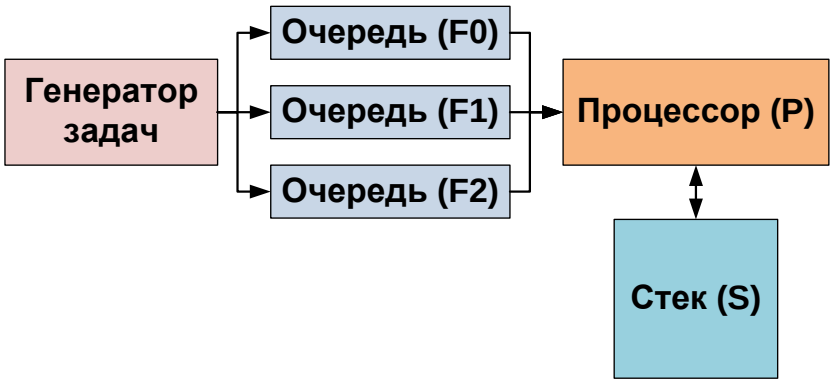
Таблица 1

№ вар.	Номер задачи	Реализация стека и очереди
1	1	Стек – статический; очередь – статическая
2	1	Стек – статический; очередь – динамическая
3	1	Стек – динамический; очередь – статическая
4	1	Стек – динамический; очередь – динамическая
5	2	Стек – статический; очередь – статическая
6	2	Стек – статический; очередь – динамическая
7	2	Стек – динамический; очередь – статическая
8	2	Стек – динамический; очередь – динамическая
9	3	Стек – статический; очередь – статическая
10	3	Стек – статический; очередь – динамическая
11	3	Стек – динамический; очередь – статическая
12	3	Стек – динамический; очередь – динамическая
13	4	Стек – статический; очередь – статическая

№ вар.	Номер задачи	Реализация стека и очереди
14	4	Стек – статический; очередь – динамическая
15	4	Стек – динамический; очередь – статическая
16	4	Стек – динамический; очередь – динамическая
17	5	Стек – статический; очередь – статическая
18	5	Стек – статический; очередь – динамическая
19	5	Стек – динамический; очередь – статическая
20	5	Стек – динамический; очередь – динамическая
21	6	Стек – статический; очередь – статическая
22	6	Стек – статический; очередь – динамическая
23	6	Стек – динамический; очередь – статическая
24	6	Стек – динамический; очередь – динамическая
25	7	Стек – статический; очередь – статическая
26	7	Стек – статический; очередь – динамическая
27	7	Стек – динамический; очередь – статическая
28	7	Стек – динамический; очередь – динамическая
29	8	Стек – статический; очередь – статическая
30	8	Стек – статический; очередь – динамическая
31	8	Стек – динамический; очередь – статическая
32	8	Стек – динамический; очередь – динамическая

Задача 1.

Система состоит из процессора *P*, трёх очередей *F*0, *F*1, *F*2 и стека *S*. В систему поступают запросы на выполнение задач.

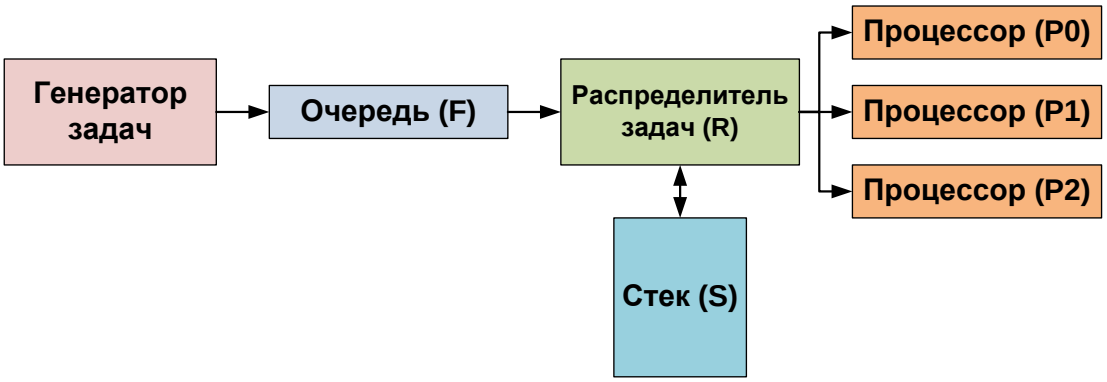


Поступающие запросы ставятся в соответствующие приоритетам очереди. Сначала обрабатываются задачи из очереди *F*0. Если она пуста, можно обрабатывать задачи из очереди *F*1. Если и она пуста, то можно обрабатывать задачи из очереди *F*2. Если все очереди пусты, то система находится в ожидании поступающих задач (процессор свободен), либо в режиме обработки предыдущей задачи (процессор занят). Если поступает задача с более высоким приоритетом, чем обрабатываемая в данный момент,

то обрабатываемая помещается в стек и может обрабатываться тогда и только тогда, когда все задачи с более высоким приоритетом уже обработаны.

Задача 2.

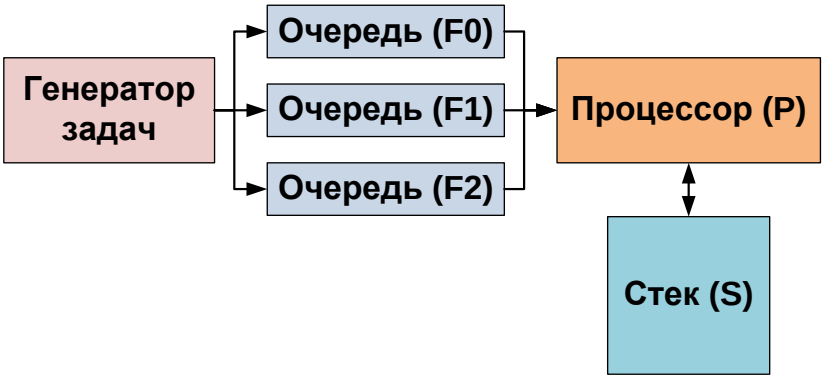
Система состоит из трех процессоров $P0$, $P1$, $P2$, очереди F, стека S и распределителя задач R. В систему поступают запросы на выполнение задач трёх типов – $T0$, $T1$ и $T2$, каждая для своего процессора.



Поступающие запросы ставятся в очередь. Если в начале очереди находится задача T_i и процессор P_i свободен, то распределитель R ставит задачу на выполнение в процессор P_i, а если процессор P_i занят, то распределитель R отправляет задачу в стек и из очереди извлекается следующая задача. Если в вершине стека находится задача, процессор которой в данный момент свободен, то эта задача извлекается и отправляется на выполнение.

Задача 3.

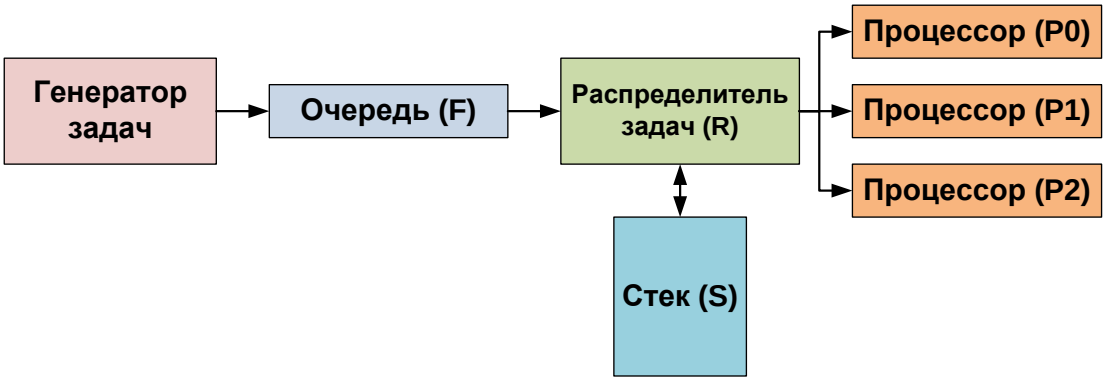
Система состоит из процессора P, трёх очередей $F0$, $F1$, $F2$ и стека S. В систему поступают запросы на выполнение задач.



Поступающие запросы ставятся в соответствующие приоритетам очереди. Сначала обрабатываются задачи из очереди $F0$. Если она пуста, можно обрабатывать задачи из очереди $F1$. Если и она пуста, то можно обрабатывать задачи из очереди $F2$. Если все очереди пусты, то система находится в ожидании поступающих задач (процессор свободен), либо в режиме обработки предыдущей задачи (процессор занят). Если поступает задача с более высоким приоритетом, чем обрабатываемая в данный момент, то обрабатываемая помещается в стек и может обрабатываться тогда и только тогда, когда все очереди пусты.

Задача 4.

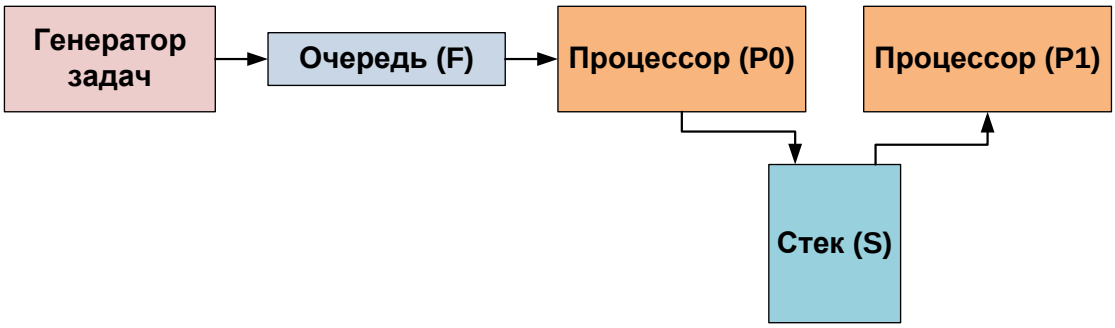
Система состоит из трех процессоров $P0$, $P1$, $P2$, очереди F, стека S и распределителя задач R. В систему поступают запросы на выполнение задач трёх типов – $T0$, $T1$ и $T2$, каждая для своего процессора.



Поступающие запросы ставятся в очередь. Если в начале очереди находится задача T_i и процессор P_i свободен, то распределитель R ставит задачу на выполнение в процессор P_i, а если процессор P_i занят, то распределитель R отправляет задачу в стек и из очереди извлекается следующая задача. Задача из стека поступает в соответствующий ей свободный процессор только тогда, когда очередь пуста.

Задача 5.

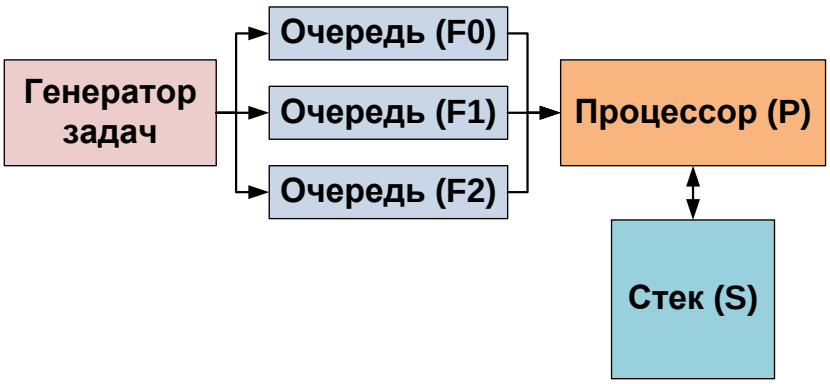
Система состоит из двух процессоров $P0$ и $P1$, стека S и очереди F. В систему могут поступать запросы на выполнение задач, причем время выполнения задачи каждым из процессоров, может отличаться. Поступающие запросы ставятся в очередь.



Если процессор P0 свободен, то в него поступает на обработку задача из очереди. После обработки задачи процессором P0, задача помещается в стек. Если стек не пустой и процессор P1 свободен, то задача извлекается из стека, и обрабатывается процессором.

Задача 6.

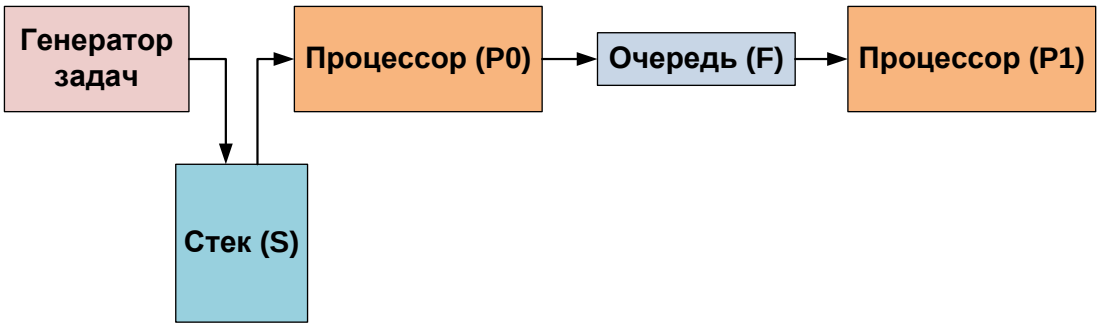
Система состоит из процессора *P*, трёх очередей *F*0, *F*1, *F*2 и стека *S*. В систему поступают запросы на выполнение задач.



Поступающие запросы ставятся в соответствующие приоритетам очереди. Сначала обрабатываются задачи из очереди *F*0. Если она пуста, можно обрабатывать задачи из очереди *F*1. Если и она пуста, то можно обрабатывать задачи из очереди *F*2. Если все очереди пусты, то система находится в ожидании поступающих задач (процессор свободен), либо в режиме обработки предыдущей задачи (процессор занят). Если поступает задача с более высоким приоритетом, чем обрабатываемая в данный момент, то обрабатываемая помещается в стек, если она выполнена менее чем на половину по времени, и может обрабатываться тогда и только тогда, когда все задачи с более высоким приоритетом уже обработаны.

Задача 7.

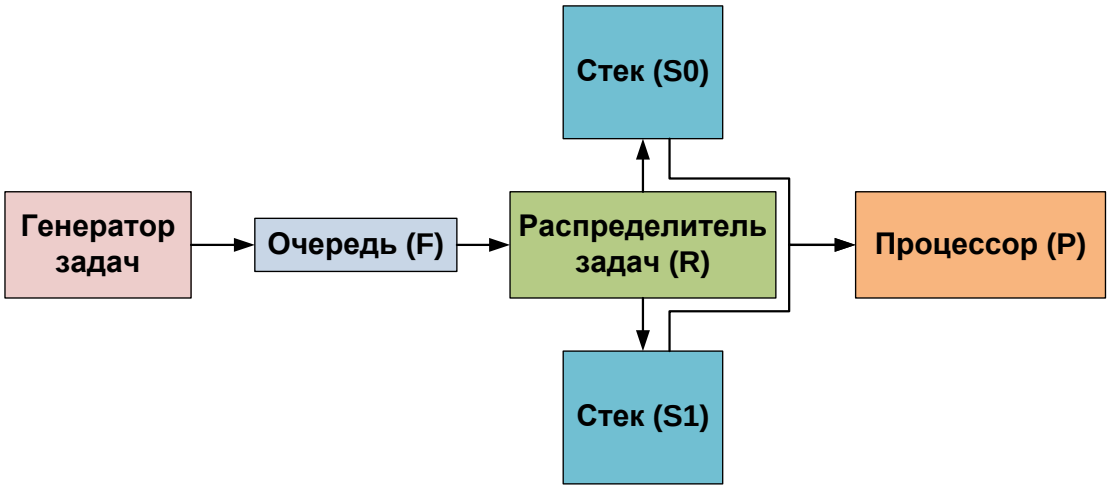
Система состоит из двух процессоров *P*0 и *P*1, стека *S* и очереди *F*. В систему могут поступать запросы на выполнение задач, причем время выполнения задачи каждым из процессоров, может отличаться. Поступающие запросы попадают в стек.



Если процессор P0 свободен, то в него поступает на обработку задача из стека. После обработки задачи процессором P0, задача помещается в очередь. Если очередь не пуста и процессор P1 свободен, то задача извлекается из очереди, и обрабатывается процессором.

Задача 8.

Система состоит из процессора *P*, двух стеков *S0* и *S1*, очереди *F* и распределителя задач *R*. Поступающие в систему запросы, попадают в очередь.



Распределитель задач (R), получает задачу из очереди и помещает ее либо в стек S0, либо в стек S1 (зависимости от приоритета задачи). Процессор P, обрабатывает задачи из стеков в порядке приоритета. Таким образом, если стек S0 пуст и процессор P свободен, то могут быть обработаны задачи из стека S1.

1.3 Порядок выполнения работы

1) выбрать вариант задания в соответствии с требованиями раздела **Ошибка! Источник ссылки не найден.**;
2) изучить теоретический материал;
3) разработать на языке программирования высокого уровня программу, выполняющую поставленную задачу с использованием заданных структур данных;
4) написать отчет о работе;
5) защитить отчет.

К защите отчета по лабораторной работе, включающую демонстрацию работы программы, необходимо сформировать два или более контрольных примера.

1.4 Содержание отчета

Отчет должен содержать:

1) титульный лист;
2) цель работы;
3) вариант задания;
4) листинг программы, реализующей поставленную задачу с использованием заданных структур данных;
5) контрольные примеры, в которых задействованы все структуры описанные в задании;
6) выводы по работе.

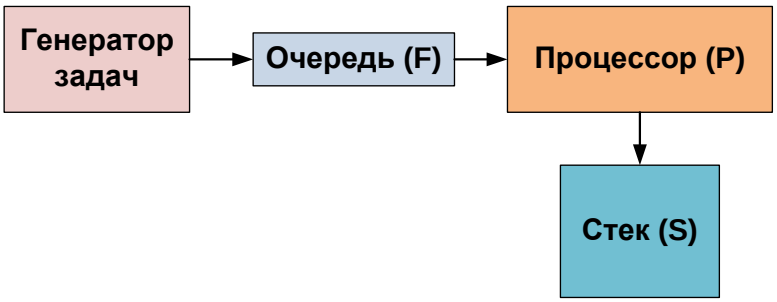
1.5 Пример выполнения работы

Предположим, что необходимо выполнить следующий вариант задания:

№ вар.	Номер задачи	Реализация стека и очереди
33	9	Стек – динамический; очередь – динамическая

Задача 9.

Система состоит из процессора *P*, очереди *F* и стека *S*. В систему поступают запросы на выполнение задач трех приоритетов.



Поступающие запросы ставятся в очередь. Если очередь пуста, то система находится в ожидании поступающих задач (процессор свободен), либо в режиме обработки предыдущей задачи (процессор занят). Если поступает задача с более высоким приоритетом, чем обрабатываемая в данный момент, то обрабатываемая помещается в стек и может обрабатываться тогда и только тогда, когда все задачи с более высоким приоритетом уже обработаны

Генератор задач должен формировать запросы на выполнение задач. В общем случае эти запросы на выполнение задач со случайным приоритетом и случайной длительности должны поступать в случайные моменты времени. Однако, для демонстрации всех свойств системы, необходимо определенная комбинация задач с заданными характеристиками. Поэтому при сдаче лабораторной студенту будет необходимо продемонстрировать работу системы на заранее запланированных примерах.

В приведенном примере, для удобства программирования, будут использованы 2 структуры, Структура, описывающая одну задачу, и структура, позволяющая хранить набор задач. В данном примере у задач нет названия, однако при выполнении лабораторной работы, каждая задача должна иметь своё название.

```
struct Task
{
	uint16_t priority;
	uint16_t taskTime;
	uint16_t durationTime;
};

struct TaskList
{
	Task *taskValues;
	TaskList *next;
};
```

Работа системы моделируется в виде последовательности тактов работы. На каждом такте выполняются следующие действия:

1) Проверяется поступление задач на текущий такт, и, если задача поступила, она ставится в очередь;
2) Проверяется приоритет ближайшей задачи из очереди, и задачи, обрабатываемой процессором. Далее, в соответствии со схемой работы системы, происходит поступление задач в стек, пока в процессоре не окажется задача с приоритетом более высоким, чем у ближайшей задачи в очереди;
3) Увеличивается таймер системы, и уменьшается время выполнения задачи, находящейся в процессоре. Если задача завершена, процессор освобождается.

Текст функции на языке C++, выполняющей данную задачу, приведен ниже:

```
void processorLoop(TaskList *&IncomingTask)
{
 TaskList *Stack = NULL;
 TaskList *Queue = NULL;
 Task *OurProcessor = new Task;
 OurProcessor->priority = 0;
 OurProcessor->durationTime = 0;
 bool emptyQueue = true; //Проверка пустоты Очереди
 bool emptyStack = true; //Проверка пустоты Стека
 bool processorIsFree = true; //Проверка занятости процессора
 bool allTasksGone = false;
 int timer = 1;
 while(true)
 {
  if (!allTasksGone)
  {
   if (IncomingTask->taskValues->taskTime == timer)
   {
    pushToQueue(IncomingTask, Queue, emptyQueue, allTasksGone);
   }
  }
  if (!emptyQueue)
  {
   if (Queue->taskValues->priority > OurProcessor->priority || processorIsFree)
   {
    if (OurProcessor->durationTime > 0)
     pushToStack(Stack, OurProcessor, emptyStack, processorIsFree);
    getFromQueue(Queue, OurProcessor, emptyQueue, processorIsFree);
    while (true)
    {
     if (!emptyQueue)
     {
```

```
      if (Queue->taskValues->priority > OurProcessor->priority)
      {
       pushToStack(Stack, OurProcessor, emptyStack, processorIsFree);
       getFromQueue(Queue, OurProcessor, emptyQueue, processorIsFree);
      }
      else
       break;
     }
     else
      break;
    }
   }
  }
  else if (!emptyStack)
  {
   if (processorIsFree)
    getFromStack(Stack, OurProcessor, emptyStack, processorIsFree);
  }
  cout << endl << "Идет " << timer << " такт" << endl;
  if (!allTasksGone)
  {
   cout << "Входные задания" << endl;
   showStruct(IncomingTask);
  }
  if (!emptyStack)
  {
   cout << "Содержимое стэка" << endl;
   showStruct(Stack);
  }
  if (!emptyQueue)
  {
   cout << "Содержимое очереди" << endl;
   showStruct(Queue);
  }
  if (!processorIsFree)
  {
   cout << "Содержимое процессора" << endl;
   showStructElem(OurProcessor);
  }
  else
   cout << "Процессор свободен" << endl;
  if (!processorIsFree)
  {
   if (OurProcessor->durationTime)
    OurProcessor->durationTime--;
   if(OurProcessor->durationTime<=0)
   {
    OurProcessor->durationTime = 0;
    OurProcessor->priority = 0;
    processorIsFree = true;
   }
  }
  timer++;
  if (emptyStack && emptyQueue && processorIsFree && allTasksGone)
   break;
 }
 IncomingTask=NULL;
}
```

Теперь рассмотрим пример работы. Составим, в качестве контрольного примера, следующее расписание поступление запросов на выполнение задач:

Момент поступления	Длительность выполнения	Приоритет
1	2	3
2	1	3
3	2	1
4	3	2
5	3	3

Результат работы программы по заданному расписанию будет следующим:

```
Идет 1 такт
Входные задания
Время поступления задачи 2 Приоритет задачи 3 Такты задачи 1
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 3
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 3
Содержимое процессора
Время поступления задачи 1 Приоритет задачи 3 Такты задачи 2

Идет 2 такт
Входные задания
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 3
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 3
Содержимое очереди
Время поступления задачи 2 Приоритет задачи 3 Такты задачи 1
Содержимое процессора
Время поступления задачи 1 Приоритет задачи 3 Такты задачи 1

Идет 3 такт
Входные задания
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 3
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 3
Содержимое очереди
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 2 Приоритет задачи 3 Такты задачи 1

Идет 4 такт
Входные задания
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 3
Содержимое стэка
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 3
```

```
Идет 5 такт
Содержимое стэка
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 2
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 3

Идет 6 такт
Содержимое стэка
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 2
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 2

Идет 7 такт
Содержимое стэка
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 2
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 5 Приоритет задачи 3 Такты задачи 1

Идет 8 такт
Содержимое стэка
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 2

Идет 9 такт
Содержимое стэка
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2
Содержимое процессора
Время поступления задачи 4 Приоритет задачи 2 Такты задачи 1

Идет 10 такт
Содержимое процессора
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 2

Идет 11 такт
Содержимое процессора
Время поступления задачи 3 Приоритет задачи 1 Такты задачи 1
```

Также следует отметить, что результаты показаны по состоянию на конец такта. Таким образом, если задача ставится в очередь и сразу попадает на выполнение, то факт ее пребывания в очереди не отображается. Не отображается и восстановление прерванной задачи из стека, и новое ее прерывание на этом же такте при наличии более приоритетных задач в очереди.

1.6 Контрольные вопросы

1) Что такое стек?
2) Назовите основные способы реализации стека. Сравните их.
3) Что такое очередь?
4) Назовите основные способы реализации очереди. Сравните их.
5) Что такое дек?
6) В каких случаях целесообразно применять стек, а каких очередь?
7) Какие достоинства и недостатки использования статических и динамических структур данных?