Лабораторная работа № 5

Тема: «Стэк и очередь»

Цель работы: Изучение стэка и очереди

Стек — это коллекция, элементы которой получают по принципу «последний вошел, первый вышел» *(Last-In-First-Out или LIFO)*. Это значит, что мы будем иметь доступ только к последнему добавленному элементу.

Очереди очень похожи на стеки. Они также не дают доступа к произвольному элементу, но, в отличие от стека, элементы кладутся *(enqueue)* и забираются *(dequeue)* с разных концов. Такой метод называется «первый вошел, первый вышел» *(First-In-First-Out или FIFO)*. То есть забирать элементы из очереди мы будем в том же порядке, что и клали. Как реальная очередь или конвейер.

```cpp
#include <iostream>
#include <string>
#include <fstream>
#include <iomanip>
using namespace std;


struct prog {
    string name;//имя
    int prioritet; //приоритет
    int vremya; //время выполнения
    int time; //время когда начать выполнять процесс
};

int add_spisok(prog *task) {
    int count;
    ifstream file_in("in_file.txt", ios:: in );

    if (!file_in) {
        cerr<<"Ошибка открытия файла in_file.txt"<<endl;
        return 0;
    }
    //Добавляем задачи в генератор задач
    for (count = 0; !file_in.eof(); count++) {
        file_in >> task[count].name;
        file_in >> task[count].prioritet;
        file_in >> task[count].vremya;
        file_in >> task[count].time;
    }
    return count;
};

void sdvig(prog *F, int &iF){
    //сдвигаем очередь на 1
    for (int i = 0; i < iF; i++)
        F[i] = F[i+1];
    iF--;
};

int main () {
    setlocale(LC_ALL,"rus");
    prog F0[100], F1[100], F2[100], //3 очереди
            stack[100], //стек
            task[300], //генератор задач
            CPU[2]; //процессоры для выполнения задач
    int iF0 = 0, iF1 = 0, iF2 = 0, iS = 0;
    int timer = 0, count = 0;
    bool newTask = false;

    //Добавляем задачи
    count = add_spisok(task);
    CPU[2].name = "";
```

```
for (int i = 0; i < count ; timer++) {
    //генератор задач
    do {
        newTask = false;
        if (task[i].time == timer) {
            if (iF0 == 100)
                iF0 = 0;
            if (iF1 == 100)
                iF1 = 0;
            if (iF2 == 100)
                iF2 = 0;
            switch (task[i].prioritet) {
                case 0: {
                    F0[iF0] = task[i];
                    iF0++;
                break;
                    }
                case 1: {F1[iF1] = task[i];
                    iF1++;
                break;
                }
                case 2: {F2[iF2] = task[i];
                    iF2++;
                break;
                }
            }
            i++;
            newTask = true;
        }
    } while (newTask);

  //Работа процессора
  if (CPU[0].name != "") {
      CPU[0].vremya--;
      if (CPU[0].vremya <= 0) {
          CPU[0].name = "";
          CPU[0].prioritet = 3;
          if (iS > 0) {
              CPU[0] = stack[iS-1];
              iS--;
          }
      }
  }
```

```
    //Обработка очереди 0
    if (iF0 > 0) {
        if (CPU[0].name != "") {
            if (CPU[0].prioritet > F0[0].prioritet) {
                stack[iS] = CPU[0];
                iS++;
                CPU[0] = F0[0];
                sdvig(F0, iF0);
            }
        } else {
            CPU[0] = F0[0];
            sdvig(F0, iF0);
        }

    }
    //Обработка очереди 1
    if (iF1 > 0) {
        if (CPU[1].name != "") {
            if (CPU[1].prioritet > F1[0].prioritet) {
                stack[iS] = CPU[1];
                iS++;
                CPU[1] = F1[0];
                sdvig(F1, iF1);
            }
        } else {
            CPU[1] = F1[0];
            sdvig(F1, iF1);
        }
    }
    //Обработка очереди 2
    if (iF2 > 0) {
        if (CPU[0].name != "") {

                if (CPU[0].prioritet > F2[0].prioritet) {
                stack[iS] = CPU[0];
                iS++;
                CPU[0] = F2[0];
                sdvig(F2, iF2);
              }
           } else {
               CPU[0] = F2[0];
               sdvig(F2, iF2);
           }
        }

        cout << "Timer " << setw(2) << timer << "  CPU=" << setw(1) << CPU[0].name;
        cout << "  \tF0=";
        for (int i = 0; i < iF0; i++) cout << F0[i].name << ",";
        cout << "\t F1=";
        for (int i = 0; i < iF1; i++) cout << F1[i].name << ",";
        cout << "\t F2=";
        for (int i = 0; i < iF2; i++) cout << F2[i].name << ",";
        cout << "    \t S=";
        for (int i = 0; i < iS; i++) cout << stack[i].name << ",";
        cout << endl;
    }

    system("pause");
    return 0;
}
```