**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

по дисциплине **«**Операционные системы»

на тему: «Простейшие схемы управления памятью»

Выполнил: студент гр. ИТП-11

Косиченко Д.А.

Принял: преподаватель-стажёр

Карась О.В.

Гомель 2022

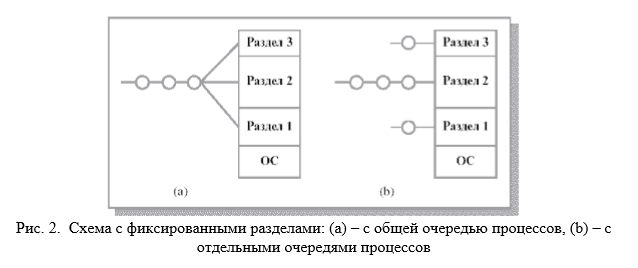
**Цель работы:** изучение алгоритмов управления памятью, разработка программы менеджера памяти.

**Схема с фиксированными разделами**

Самым простым способом управления оперативной памятью является ее предварительное (обычно на этапе генерации или в момент загрузки системы) разбиение на несколько разделов фиксированной величины. Поступающие процессы помещаются в тот или иной раздел. При этом происходит условное разбиение физического адресного пространства. Связывание логических и физических адресов процесса происходит на этапе его загрузки в конкретный раздел, иногда – на этапе компиляции.

Каждый раздел может иметь свою очередь процессов, а может существовать и глобальная очередь для всех разделов (см. рис. 2).

Подсистема управления памятью оценивает размер поступившего процесса, выбирает подходящий для него раздел, осуществляет загрузку процесса в этот раздел и настройку адресов.



Очевидный недостаток этой схемы – число одновременно выполняемых процессов ограничено числом разделов.

Другим существенным недостатком является то, что предлагаемая схема сильно страдает от внутренней фрагментации – потери части памяти, выделенной процессу, но не используемой им. Фрагментация возникает потому, что процесс не полностью занимает выделенный ему раздел или потому, что некоторые разделы слишком малы для выполняемых пользовательских программ.

**Задание**

Разработать программу, реализующую заданный алгоритм выделения памяти.

Менеджер памяти должен:

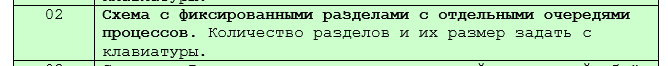
1. По запросу процесса выделять память, согласно заданного алгоритма (таблица). На экран должна выводиться следующая информация о состоянии памяти: объем памяти, объём свободной памяти, размер наибольшего свободного блока, количество запросов на выделение памяти, количество удовлетворённых запросов (%).

2. Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «ДОБАВИТЬ» память выделяется или выдаѐтся сообщение о невозможности выделения.

3. Удалять из памяти заданный блок или все блоки заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемого блока и имя процесса.

4. Реализовать возможность последовательной записи/чтения информации в/из выделенную память по логическому адресу. Вывести физического адреса ячейки памяти, в которую была осуществлена запись.

5. Организовывать циклическое выделение и освобождение памяти. При этом случайным образом задается количество выделяемых блоков и их размер.



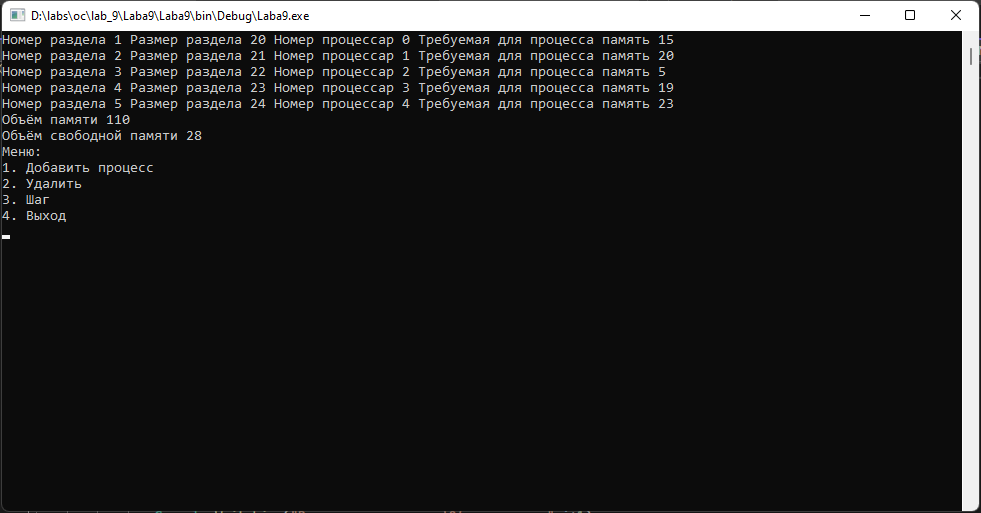


Рисунок 1 – Ввод необходимой информации

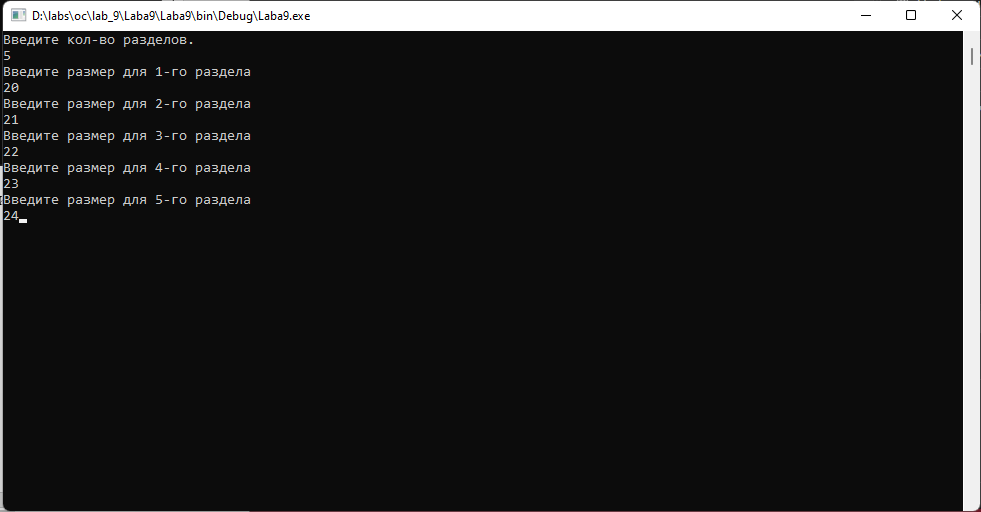


Рисунок 2 – Выполнение процессов

Листинг программы

using System;

using System.IO;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Laba9

{

class Program

{

static void Main(string[] args)

{

int n, i, amount = 0, amount1 = 0, k = 0, j = 0, f = 0, ff = 0, t = 0, del;

Console.WriteLine("Введите кол-во разделов.");

n = Convert.ToInt32(Console.ReadLine());

int[,] A = new int[n,3];

int[,] B = new int[3, 100];

for (i=0; i< n; i++)

{

Console.WriteLine("Введите размер для {0}-го раздела",i+1);

A[i, 0] = Convert.ToInt32(Console.ReadLine());

amount = amount + A[i, 0];

}

Console.Clear();

while (k != 4)

{

Console.WriteLine("Меню:");

Console.WriteLine("1. Добавить процесс");

Console.WriteLine("2. Удалить");

Console.WriteLine("3. Шаг");

Console.WriteLine("4. Выход");

try

{

k = Convert.ToInt32(Console.ReadLine());

}

catch (FormatException)

{

k = 3;

}

switch (k)

{

case 1:

{

B[0, j] = 0;

Console.WriteLine("Введите количество требуемой процессу памяти");

B[1, j] = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите в какой раздел добавить");

B[2, j] = Convert.ToInt32(Console.ReadLine());

j++;

Console.Clear();

break;

}

case 2:

{

Console.Write(" Нажмите номер процесса которого вы хотите удалить");

try

{

del = Convert.ToInt32(Console.ReadLine());

try

{

if (B[1, del] != 0 && B[2, del] != 0)

{

B[1, del] = 0;

B[2, del] = 0;

Console.Clear();

Console.WriteLine("Процесс удалён");

}

else

{

Console.Clear();

Console.WriteLine("Такого процесса нет");

}

}

catch (IndexOutOfRangeException)

{

Console.Clear();

Console.WriteLine("Такого процесса нет");

}

}

catch(FormatException)

{

Console.Clear();

}

break;

}

case 3:

{

Console.Clear();

while (f < n)

{

while(ff == 0 && t < 100)

{

if (B[2,t] == f+1 )

{

Console.Write("Номер раздела {0} Размер раздела {1} Номер процессар {2} Требуемая для процесса память {3} ", f+1, A[f,0],t, B[1,t]);

if(B[1,t] > A[f, 0])

{

Console.Write(" Отказано!!!");

B[1, t] = 0;

}

Console.WriteLine();

amount1 = amount1 + B[1, t];

B[2, t] = 0;

ff = 1;

}

t++;

if (ff == 0 && t == 100)

{

Console.WriteLine("Раздел {0} размер {1} нужно памяти для р 0", f + 1, A[f, 0]);

}

}

t = 0;

ff = 0;

f++;

}

f = 0;

Console.WriteLine("Объём памяти {0}",amount);

Console.WriteLine("Объём свободной памяти {0}", amount-amount1);

amount1 = 0;

break;

}

case 4:

{

break;

}

}

}

Console.ReadKey();

}

}

}

**Вывод:** В ходе выполнения лабораторной работы были изучены алгоритмы управления памятью и разработана программа менеджера памяти.