**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

дисциплина «Операционные системы»

«Простейшие схемы управления памятью.»

Выполнил: студент гр.ИТП-11

Ермолин Д.К.

Принял: преподаватель

Карась О.В.

Гомель 2022

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

**Задание:** Разработать программу, реализующую заданный алгоритм выделения памяти. Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданному алгоритма. На экран должна выводится следующая информация о состоянии памяти: объём 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 выдаётся сообщение о невозможности выделения.

3. Удалять из памяти заданный блок или все блоки заданного процесса (по нажатию на кнопку “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. Организовать циклическое выделение и освобождение памяти. При этом случайным образом задаётся количество выделяемых блоков и их размер.

Вариант 11: Схема с фиксированными разделами с отдельными очередями процессов. Количество разделов и их размер задать с клавиатуры.

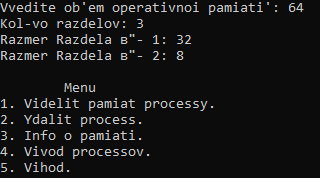


Рисунок 1 – Ввод данных

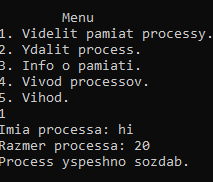


Рисунок 2 – Создание 1 процесса

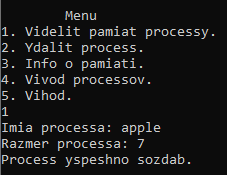


Рисунок 3 – Создание 2 процесса

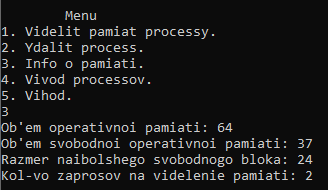


Рисунок 4 – Информация о процессе

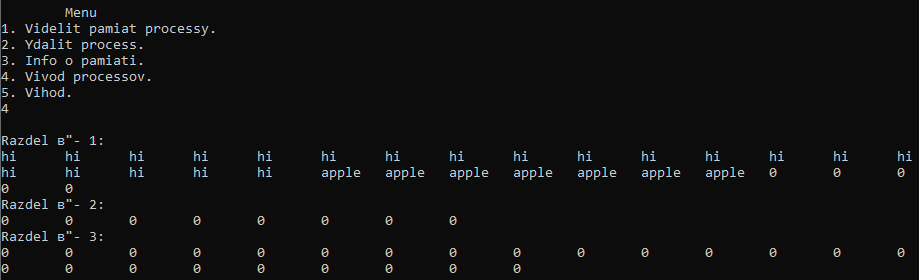


Рисунок 5 – Вывод процессов

**Вывод:** в данной лабораторной работе была разработана программа, реализующая заданный алгоритм выделения памяти.

**Приложение А**

**Листинг программы:**

#include <stdio.h>

#include <locale.h>

#include <string.h>

#include <conio.h>

#define LEN 100

struct Process

{

char name[LEN]; // Имя процесса в памяти (если процесс не создан, то по умолчанию имя "0".

int size; // Размер процесса (изначально раздела).

int num; // Номер раздела по порядку.

};

void memory(Process\* arr, int size, int& kol, int count); // Функция разделения памяти на заданное кол-во

void print\_proc(Process\* arr, int size, int kol, int count); // Вывод процессов на экран.

void menu(Process\* arr, int size, int kol, int count); // Меню.

void memory\_info(Process\* arr, int size, int kol, int count); // Вывод информации о состоянии памяти.

void delete\_proc(Process\* arr, int size, int kol, int count); // Удаление процесса.

void allocate\_mem(Process\* arr, int size, int kol, int& count); // Добавление процесса в память.

int main()

{

setlocale(LC\_ALL, "Rus");

int kol; // Количество разделов оперативной памяти.

int size; // Размер массива структур (объём оперативной памяти).

int count = 0; // Количество запросов на выделение памяти.

bool flag = true;

while(flag){

printf("Vvedite ob'em operativnoi pamiati': ");

if((scanf("%d",&size)) && (size > 0))

flag = false;

else

{

fflush(stdin);

puts("Ne chislo\n");

}

}

Process\* fix = new Process[size]; // Массив структур.

for (int i = 0; i < size; i++)

{

fix[i].size = size;

strcpy(fix[i].name, "0");

}

memory(fix, size, kol, count); // Вызов функции разделения памяти на разделы.

delete[] fix; // Очистка памяти.

getch();

return(0);

}

void memory(Process \*arr, int size, int& kol, int count)

{

int f = 0, j = 0;

int x = 1;

int s;

bool flag = true;

while(flag){

printf("Kol-vo razdelov: ");

if((scanf("%d", &kol)) && (kol > 0))

flag = false;

else

{

fflush(stdin);

puts("Nevernie dannie.Vvedite zanovo!\n");

}

}

if (kol != 1)

{

do

{

printf("Razmer Razdela № %d: ", x);

scanf("%d", &s);

if (arr[f].size > s)

{

f += s;

while (j < f)

{

arr[j].size = s;

arr[j].num = x;

j++;

}

x++;

while (j < size)

{

arr[j].size -= s;

arr[j].num = x;

j++;

}

j = f;

}

else

printf("Ne ydaetsia videlit pamiat pod razdel.\nPovtorite popitky.\n");

} while (x < kol);

}

else

{

for (int i = 0; i < size; i++)

arr[i].num = kol;

}

menu(arr, size, kol, count);

}

void print\_proc(Process\* arr, int size, int kol, int count)

{

printf("\n");

for (int i = 1; i <= kol; i++)

{

printf("Razdel № %d:\n", i);

for (int j = 0; j < size; j++)

{

if (arr[j].num == i)

printf("%s\t", arr[j].name);

}

printf("\n");

}

menu(arr, size, kol, count);

}

void menu(Process\* arr, int size, int kol, int count)

{

int question;

puts("\n\tMenu");

puts("1. Videlit pamiat processy.");

puts("2. Ydalit process.");

puts("3. Info o pamiati.");

puts("4. Vivod processov.");

puts("5. Vihod.");

scanf("%d", &question);

switch (question)

{

case 1:

allocate\_mem(arr, size, kol, count);

break;

case 2:

delete\_proc(arr, size, kol, count);

break;

case 3:

memory\_info(arr, size, kol, count);

break;

case 4:

print\_proc(arr, size, kol, count);

break;

case 5:

puts("Vipolnenie programmi zaversheno!");

break;

default:

puts("Takogo pynkta v menu net.");

menu(arr, size, kol, count);

break;

}

}

void memory\_info(Process\* arr, int size, int kol, int count)

{

int x = 0, free = 0, y;

printf("Ob'em operativnoi pamiati: %d\n", size);

for (int i = 0; i < size; i++)

{

if (strcmp("0", arr[i].name) == 0)

free++;

}

printf("Ob'em svobodnoi operativnoi pamiati: %d\n", free);

if (free <= size)

{

for (y = 0; y < size; y++)

{

if (strcmp("0", arr[y].name) == 0)

{

x = arr[y].size;

break;

}

}

while (y < size)

{

if (strcmp("0", arr[y].name) == 0 && arr[y].size > x)

x = arr[y].size;

y++;

}

printf("Razmer naibolshego svobodnogo bloka: %d\n", x);

}

printf("Kol-vo zaprosov na videlenie pamiati: %d\n", count);

menu(arr, size, kol, count);

}

void delete\_proc(Process\* arr, int size, int kol, int count)

{

fflush(stdin);

char name[LEN];

int i, num, j, k = 0;

bool key = false;

printf("Vvedite imia processa: ");

gets(name);

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

{

if (strcmp(arr[i].name, name) == 0)

{

num = arr[i].num;

key = true;

j = i - 1;

break;

}

}

if (key)

{

while ((arr[i].num == num && strcmp(arr[i].name, name) == 0 || arr[i].num == num &&

strcmp(arr[i].name, "0") == 0) && i < size)

{

strcpy(arr[i].name, "0");

k++;

i++;

}

while (j >= 0 && strcmp(arr[j].name, "0") == 0 && arr[j].num == num)

{

k++;

j--;

}

j++;

while (strcmp(arr[j].name, "0") == 0 && arr[j].num == num && j < size)

{

arr[j].size = k;

j++;

}

printf("Process %s yspeshno ydalen.\n", name);

}

else

printf("Processa %s ne syshestvyet.\n", name);

menu(arr, size, kol, count);

}

void allocate\_mem(Process\* arr, int size, int kol, int& count)

{

fflush(stdin);

count++;

char name[LEN];

int sz, num, i, free, f;

bool key = false;

printf("Imia processa: ");

gets(name);

printf("Razmer processa: ");

scanf("%d", &sz);

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

{

if (arr[i].size >= sz && strcmp(arr[i].name, "0") == 0)

{

//printf("OK");

num = arr[i].num;

free = arr[i].size;

f = free;

break;

}

}

while (strcmp(arr[i].name, "0") == 0 && arr[i].num == num)

{

strcpy(arr[i].name, name);

arr[i].size = sz;

f--;

i++;

if (f == free - sz)

{

key = true;

break;

}

}

while (i < size && strcmp(arr[i].name, "0") == 0 && arr[i].num == num)

{

arr[i].size -= sz;

i++;

}

if (key)

puts("Process yspeshno sozdab.");

else

puts("Nevozmozhno videlit pamiat processy.");

menu(arr, size, kol, count);}