Федеральное государственное автономное

образовательное учреждение

Высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| институт |
| Информатика |
| кафедра |

**ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №3**

|  |
| --- |
| Использование механизма виртуальной памяти в программах для OC GNU/LINUX |
| Тема |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Преподаватель | |  |  |  | А. Н. Пупков |
|  | |  | подпись, дата |  | инициалы, фамилия |
| Студент | КИ21-16/1б, 032155832 |  |  |  | Н. А. Терентьев |
|  | номер группы, зачётной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2023

**СОДЕРЖАНИЕ**

1 Цели 3

2 Задачи 3

3 Описание варианта задания 3

4 Ход выполнения 3

4.1 Структуры данных 3

4.2 Основные функции 4

4.3 Doxygen документация 5

4.4 Проверка на утечки памяти 6

5 Выводы 7

Список использованных источников 8

Приложение А 9

**1 Цель**

Изучение механизма виртуальной памяти в программах для OC.

**2 Задачи**

Для выполнения практической работы необходимо выполнить  
следующие задачи:

* ознакомиться с теоретическим материалом по управлению областями виртуальной памяти в ОС GNU/Linux;
* получить у преподавателя собственный вариант задания, который предусматривает моделирование и реализацию некоторой абстрактной структуры данных;
* разработать программу в соответствии с полученным заданием, в которой должен использоваться механизм управления кучами либо стеком. Обеспечить сборку программы с использованием утилиты GNU make;
* составить общее описание результатов, инструкции по сборке и использованию программ, а также инструкцию по получению документации, сформировать архив формата tar.gz и представить на проверку с исходными текстами программы.

**3 Описание варианта задания**

Работа выполнена в соответствии с вариантом 9.

Структура данных: предприятие-изготовитель; тип; цена; производительность. Создать два запроса, позволяющих найти список предприятий с наилучшим соотношением цена/производительность и диапазон цен.

**4 Ход выполнения**

**4.1 Структуры данных**

В файле structure.h были написаны структуры, необходимые для выполнения работы. Код данных структур представлены в листинге 1.

Листинг 1 – Содержимое файла structure.h

typedef struct Enterprise Enterprise; //!< Структура предприятия

typedef struct Node Node; //!< Структура списка

struct Enterprise

{

char\* company; /\*!< Предприятие изготовитель \*/

char\* type; /\*!< Тип \*/

float price; /\*!< Цена \*/

float performance; /\*!< Производительность \*/

};

struct Node

{

Enterprise data; /\*!< Данные \*/

struct Node\* next; /\*!< Указатель на следующий элемент \*/

};

**4.2 Основные функции**

В файле functions.c были реализованы функции по добавлению одного элемента списка, модификации элемента списка, удаления элемента списка, чтения одного элемента списка, отображения содержимого всех элементов списка (или его части). Также были написаны две функции для нахождения предприятий с наилучшим соотношением цена/производительность и диапазон цен. Работа с программой представлена на рисунке 1.

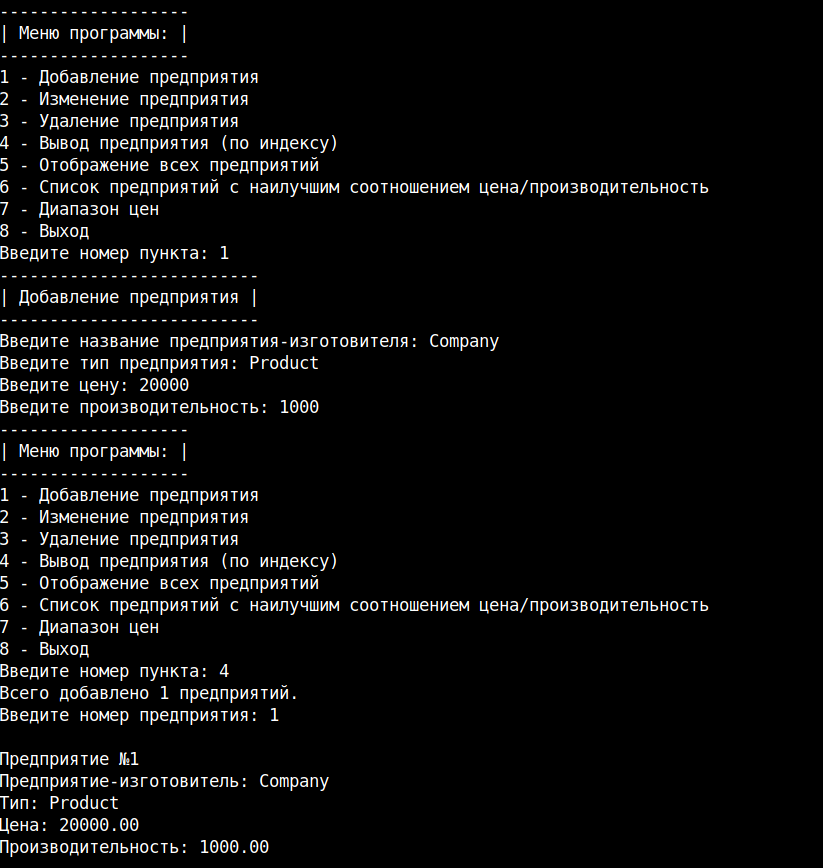


Рисунок 1 – Работа с программой

**4.3 Doxygen документация**

Все файлы содержат комментарии в стиле системы doxygen. Благодаря конфигурационному файлу была сгенерирована документация. Её внешний вид представлен на рисунке 2.

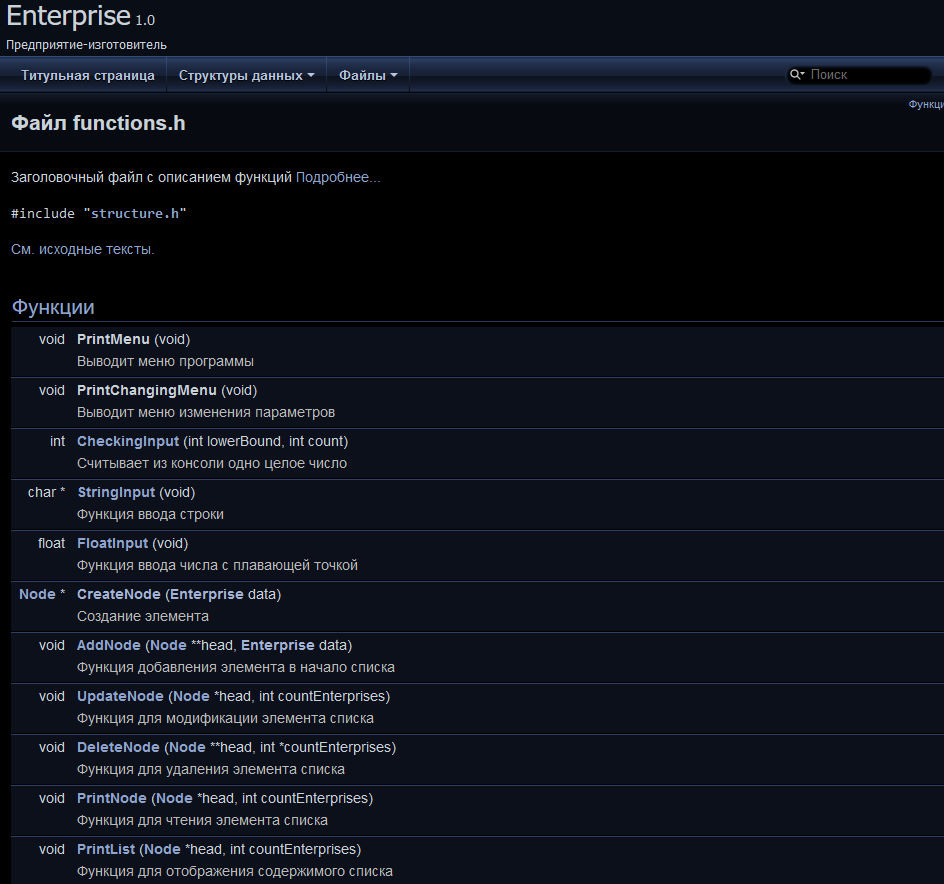


Рисунок 2 – Сгенерированная документация

**4.4 Проверка на утечки памяти**

Воспользуемся утилитой valgrind, чтобы проверить нашу программу на утечки памяти. Вывод утилиты позволяет понять, что утечек памяти нет. На рисунке 3 представлен результат работы программы.

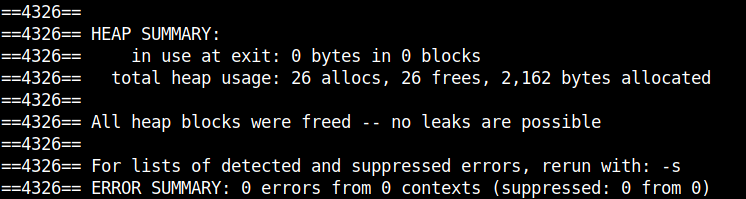


Рисунок 2 – Утилита valgrind

**5** **Выводы**

В ходе работы был изучен механизм виртуальной памяти в программах для ОС GNU/Linux. Была разработана программа в соответствии с вариантом.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 СТО 4.2-07-2014 Система менеджмента качества. Общие требования  
к построению, изложении и оформлению документов учебной деятельности.  
Дата введения – 30.12.2-13.

**ПРИЛОЖЕНИЕ А**

**Исходный код файла functions.c**

#include <stdlib.h>

#include <stdio.h>

#include <stdbool.h>

#include "structure.h"

#define CORRECT\_INPUT 2

#define END\_STRING '\n'

#define BACKSPACE\_KEY 8

#define START\_CHAR\_RANGE 32

#define END\_CHAR\_RANGE 126

void PrintMenu(void)

{

printf("-------------------\n");

printf("| Меню программы: |\n");

printf("-------------------\n");

printf("1 - Добавление предприятия\n");

printf("2 - Изменение предприятия\n");

printf("3 - Удаление предприятия\n");

printf("4 - Вывод предприятия (по индексу)\n");

printf("5 - Отображение всех предприятий\n");

printf("6 - Cписок предприятий с наилучшим соотношением "

"цена/производительность\n");

printf("7 - Диапазон цен\n");

printf("8 - Выход\n");

}

void PrintChangingMenu(void)

{

printf("--------------\n");

printf("| Параметры: |\n");

printf("--------------\n");

printf("1 - Предприятие-изготовитель\n");

printf("2 - Тип\n");

printf("3 - Цена\n");

printf("4 - Производительность\n");

}

int CheckingInput(int lowerBound, int count)

{

int userInput = 1;

char inputChar = '\0';

int input = scanf("%d%c", &userInput, &inputChar);

if (count)

{

while (input != CORRECT\_INPUT || inputChar != END\_STRING

|| userInput < lowerBound || userInput > count)

{

if (userInput >= lowerBound && userInput <= count)

{

while ((inputChar = getchar()) != '\n');

}

userInput = 1;

printf("Неверный ввод. Попробуйте снова.\nВведите номер: ");

input = scanf("%d%c", &userInput, &inputChar);

}

}

else

{

while (input != CORRECT\_INPUT || inputChar != END\_STRING

|| userInput <= lowerBound)

{

if (userInput > lowerBound)

{

while ((inputChar = getchar()) != '\n');

}

userInput = 1;

printf("Неверный ввод. Попробуйте снова.\nВведите номер: ");

input = scanf("%d%c", &userInput, &inputChar);

}

}

return userInput;

}

char\* StringInput(void)

{

char\* userStr = (char\*)malloc(1 \* sizeof(char));

userStr[0] = '\0';

char curChar = 0;

int curSize = 1;

while(curChar != '\n')

{

curChar = getchar();

int deltaVal = 0; // Определяет, на сколько изменится длина массива

int lengthDif = 0;

// Если мы стираем символы, а не пишем их,

if (curChar == BACKSPACE\_KEY)

{

deltaVal = -1; // то будем уменьшать длину массива

lengthDif = 1; // и копировать строку до предпоследнего символа

}

// Иначе проверяем, входит ли введённый символ в диапазон печатных

else

{

if (curChar >= START\_CHAR\_RANGE && curChar <= END\_CHAR\_RANGE)

{

deltaVal = 1; // Если да, то будем увеличивать длину на 1

lengthDif = 2; // Не заполняем последние 2 символа -

// оставляем место для введённого символа и \0

}

else

{

continue; // Если это не печатный символ, то пропускаем его

}

}

// Если стирать больше нечего, но пользователь

// всё равно жмёт Backspace.

int newSize = curSize + deltaVal;

if (newSize == 0)

{

continue; // то мы переходим на следующую итерацию - ждём '\n'

}

char\* tmpStr = (char\*)malloc(newSize \* sizeof(char));

if (tmpStr) // Проверяем, выделилась ли память

{

// Идём до предпоследнего символа, т.к. надо в конец записать '\0'

for (int i = 0;

i < newSize - lengthDif;

++i)

{

tmpStr[i] = userStr[i];

}

if (curChar != BACKSPACE\_KEY) // Если введён печатный символ,

{

tmpStr[newSize - 2] = curChar; // Добавляем его в строку

tmpStr[newSize - 1] = '\0';

}

free(userStr);

userStr = tmpStr;

curSize = newSize;

}

else

{

printf("Не могу выделить память под обновлённую строку!");

break;

}

}

return userStr;

}

float FloatInput(void)

{

float number = 0;

while (number == 0 || number < 0)

{

char\* InputValue = StringInput();

number = strtof(InputValue, NULL);

if (number == 0 || number < 0)

{

printf("Неверный ввод. Попробуйте снова.\n"

"Повторите ввод: ");

}

free(InputValue);

}

return number;

}

Node\* CreateNode(Enterprise data)

{

Node\* newNode = malloc(sizeof(Node)); // Выделение памяти

if (newNode == NULL)

{ // Проверка на ошибку

printf("Не удалось выделить память.\n");

exit(1);

}

newNode->data = data; // Копирование данных

newNode->next = NULL; // Инициализация указателя

return newNode;

}

void AddNode(Node\*\* head, Enterprise data)

{

Node\* newNode = CreateNode(data); // Создание нового элемента

if (\*head == NULL) // Если список пустой

{

\*head = newNode; // Новый элемент становится головой списка

}

else // Иначе

{

Node\* temp = \*head; // Создание временного указателя temp

while (temp->next != NULL) // Перебор списка до последнего элемента

{

temp = temp->next;

}

temp->next = newNode; // Связывание последнего элемента с новым

}

}

void UpdateNode(Node\* head, int countEnterprises)

{

printf("Всего добавлено %d предприятий.\n", countEnterprises);

printf("Введите номер предприятия, которое вы хотите изменить: ");

int index = CheckingInput(1, countEnterprises) - 1;

int count = 0; // счетчик

PrintChangingMenu();

printf("Введите номер пункта: ");

int userChoice = CheckingInput(1, 4);

char\* newCompany = NULL;

char\* newType = NULL;

float newPrice = 0;

float newPerformance = 0;

if (userChoice == 1)

{

printf("Введите новое название предприятия-изготовителя: ");

newCompany = StringInput();

}

if (userChoice == 2)

{

printf("Введите новый тип предприятия: ");

newType = StringInput();

}

if (userChoice == 3)

{

printf("Введите новую цену: ");

newPrice = FloatInput();

}

if (userChoice == 4)

{

printf("Введите новую производительность: ");

newPerformance = FloatInput();

}

while (head != NULL)

{ // Пока не достигнут конец списка

if (count == index)

{ // Если счетчик совпадает с индексом

// Проверить, задано ли новое значение для каждого поля

if (newCompany != NULL)

{

free(head->data.company);

head->data.company = newCompany;

}

if (newType != NULL)

{

free(head->data.type);

head->data.type = newType;

}

if (newPrice > 0.0)

{

head->data.price = newPrice;

}

if (newPerformance > 0.0)

{

head->data.performance = newPerformance;

}

return;

}

count++;

head = head->next;

}

printf("Индекс за пределами диапазона.\n");

}

void DeleteNode(Node\*\* head, int\* countEnterprises)

{

printf("Всего добавлено %d предприятий.\n", \*countEnterprises);

printf("Введите номер предприятия, которое вы хотите удалить: ");

int index = CheckingInput(1, \*countEnterprises) - 1;

Node\* current = \*head; // текущий элемент

Node\* prev = NULL; // предыдущий элемент

int count = 0; // счетчик

while (current != NULL)

{ // Пока не достигнут конец списка

if (count == index)

{ // Если индекс совпадает с счетчиком

if (prev == NULL)

{ // Если удаляем голову списка

\*head = current->next; // Обновляем голову списка

}

else

{ // Иначе

prev->next = current->next; // Обойти удаляемый элемент

}

// Освобождаем память

free(current->data.company);

free(current->data.type);

free(current);

\*countEnterprises = \*countEnterprises - 1;

return; // Завершаем функцию

}

count++; // Увеличиваем счётчик

prev = current; // Обновляем предыдущий элемент

current = current->next; // Переходим к следующему элементу

}

printf("Индекс за пределами диапазона.\n"); // Сообщаем об ошибке

}

void PrintNode(Node\* head, int countEnterprises)

{

printf("Всего добавлено %d предприятий.\n", countEnterprises);

printf("Введите номер предприятия: ");

int index = CheckingInput(1, countEnterprises) - 1;

int count = 0; // Счетчик

bool flag = false;

while (head != NULL)

{ // Пока не достигнут конец списка

if (count == index)

{ // Если счетчик совпадает с индексом

flag = true;

break;

}

count++; // Увеличиваем счетчик

head = head->next; // Переходим к следующему элементу

}

if (flag)

{

printf("\n");

printf("Предприятие №%d\n", index + 1);

printf("Предприятие-изготовитель: %s\n", head->data.company);

printf("Тип: %s\n", head->data.type);

printf("Цена: %.2f\n", head->data.price);

printf("Производительность: %.2f\n", head->data.performance);

printf("\n");

}

else

{

printf("Индекс за пределами диапазона.\n"); // Ошибка

}

}

void PrintList(Node\* head, int countEnterprises)

{

printf("Всего добавлено %d предприятий.\n", countEnterprises);

printf("Введите начальный номер: ");

int start = CheckingInput(1, countEnterprises) - 1;

printf("Введите конечный номер: ");

int end = CheckingInput(start, countEnterprises) - 1;

int count = 0; // Счетчик

while (head != NULL)

{ // Пока не достигнут конец списка

if (count >= start && count <= end)

{

// Вывести данные текущего элемента

printf("\n");

printf("Предприятие №%d\n", count + 1);

printf("Предприятие-изготовитель: %s\n", head->data.company);

printf("Тип: %s\n", head->data.type);

printf("Цена: %.2f\n", head->data.price);

printf("Производительность: %.2f\n", head->data.performance);

printf("\n");

}

count++; // Увеличить счетчик

head = head->next; // Переходим к следующему элементу

}

}

void FindBestRatio(Node\* head)

{

Node\* current = head; // Текущий элемент

double bestRatio = 0.0; // Лучшее соотношение

while (current != NULL) { // Пока не достигнут конец списка

double ratio = current->data.price /

current->data.performance; // Вычисление соотношения

if (ratio < bestRatio || bestRatio ==

0.0)

{ // Если соотношение лучше или еще не определено

bestRatio = ratio; // Обновляем лучшее соотношение

}

current = current->next; // Переходим к следующему элементу

}

printf("Лучшее соотношение цена/производительность: %.2f\n",

bestRatio);

printf("Предприятия с таким коэффициентом:\n");

current = head; // СБрасываем текущий элемент

while (current != NULL)

{ // Пока не достигнут конец списка

double ratio = current->data.price /

current->data.performance;

if (ratio == bestRatio)

{ // Если соотношение совпадает с лучшим

printf("%s\n",

current->data.company); // Выводим название предприятия

}

current = current->next; // Переходим к следующему элементу

}

}

void FindPriceRange(Node\* head) {

double minPrice = 0.0; // Минимальная цена

double maxPrice = 0.0; // Максимальная цена

while (head != NULL)

{

double price = head->data.price;

if (price < minPrice ||

minPrice == 0.0)

{

minPrice = price;

}

if (price > maxPrice ||

maxPrice == 0.0)

{

maxPrice = price;

}

head = head->next;

}

printf("Диапазон цен от %.2f до %.2f\n", minPrice, maxPrice);

}

void FreeList(Node\*\* head) {

Node\* current = \*head;

Node\* next;

while (current != NULL)

{

next = current->next;

free(current->data.company);

free(current->data.type);

free(current);

current = next;

}

\*head = NULL;

}