|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 8** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Алгоритмические стратегии или методы разработки алгоритмов. Перебор и методы его сокращения.»** | |
|  | |
| Выполнил студент группы ИКБО-10-21 | Черномуров С.А. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2022

# **Цель работы**

Получить навыки применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.

# **Постановка задачи**

**Задание.**

1. Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.
2. Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы.
3. Привести анализ снижения числа переборов при применении метода.
4. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №27. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | **Задача:**  Имеется определенный набор ингредиентов И1, И2,..., Иn (каждый в единственном экземпляре); известны их калорийность *q1, q2, ..., qn* и жирность *с1*, *с2, ..., сn.* Требуемая калорийность салата равна *Q.* Требуется приготовить салат, чтобы суммарная жирность была максимальна, при суммарной калорийности ≤Q.  **Метод:**  Жадный алгоритм |

# **Решение**

**Объяснение алгоритмов применяемых функций**

**Задание 1**

Конструктор BackpackProblem получает на вход массивы жирностей и калорий, максимальное число калорий и размер массивов и записывает их в соответствующие поля объекта класса BackpackProblem.

|  |
| --- |
| //Конструктор класса BackpackProblem  BackpackProblem::BackpackProblem(int\* calories, int\* fat, int maxCalories, int size) {  this->calories = calories;  this->fat = fat;  this->maxCalories = maxCalories;  this->size = size;  } |

Метод solve сортирует по невозрастанию по удельной жирности (жирность/калорийность) массивы ингредиентов, а дальше с помощью жадного алгоритма собирает салат, жирность которого максимальна, а количество калорий минимально, записывает нужные ингредиенты в вектор values, и возвращает кортеж, содержащий набранное число жирности, число калорий и вектор, содержащий ингредиенты салата.

|  |
| --- |
| //Метод решения задачи  tuple<int, int, vector<pair<int,int>>> BackpackProblem::solve() {  pair<int, int>\* arr = new pair<int, int>[size];  for (int i = 0; i < size; i++) {  arr[i].first = fat[i];  arr[i].second = calories[i];  }  sort(arr, arr + size, [](pair<int, int> a, pair<int, int> b) {return a.first \* 1.0 / a.second > b.first \* 1.0 / b.second; });  int currentCalories = 0;  int currentFat = 0;  vector<pair<int, int>> values;  for (int i = 0; i < size; i++) {  if (currentCalories + arr[i].second <= maxCalories) {  currentCalories += arr[i].second;  currentFat += arr[i].first;  values.push\_back(make\_pair(arr[i].first, arr[i].second));  }  }  delete[] arr;  //cout << currentCalories << " " << currentFat;  return make\_tuple(currentFat, currentCalories, values);  } |

**Описание работы пользовательского интерфейса**

Сначала выводится выбор задания, пользователь вводом с клавиатуры выбирает задание. После чего консоль очищается и отображается ввод в зависимости от выбранного задания. Введенные данные обрабатываются, результат обработки выводится на экран, программа автоматически перезапускается.

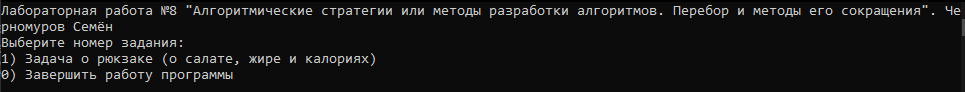


Рисунок 1. Интерфейс программы

# **Тестирование**

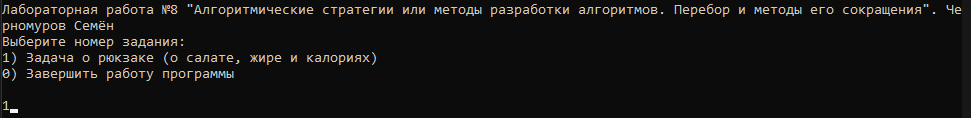


Рисунок 2. Выбор задания 1

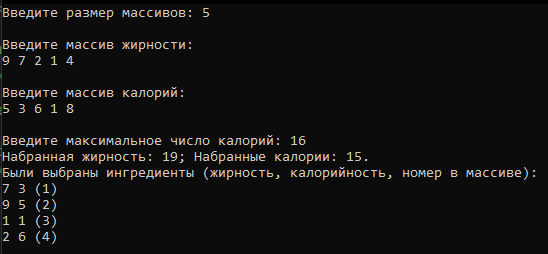


Рисунок 3. Результат работы программы (задание 1)

Программа получает на вход размеры массивов, массив жирности, массив калорийности и максимальное число калорий, далее массивы сортируются по невозрастанию по удельной жирности (порядок после сортировки: 7/3, 9/5, 1/1, 4/8, 2/6). После чего по порядку из этого массива выбираются элементы, аккумулируются суммарные жирность и калорийность до тех пор, пока суммарная калорийность меньше или равна максимальной, и добавленные ингредиенты запоминаются.

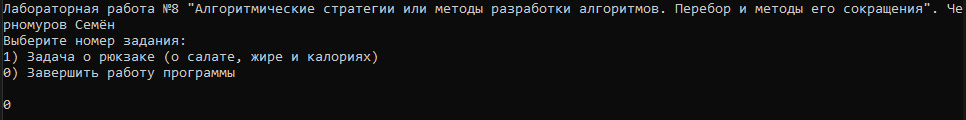


Рисунок 4. Выбор завершения работы программы

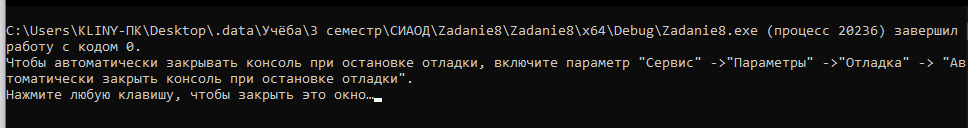


Рисунок 5. Завершение работы программы

При выборе функции завершения работы программы программа завершает свою работу.

Из результатов выполнения программы видно, что программа работает корректно.

**Оценка эффективности жадного алгоритма**

Чтобы оценить эффективность жадного алгоритма для решения подобных задач, сначала оценим число действий при поиске решения полным перебором. Это можно сделать с помощью комбинаторики. Каждый элемент можно брать или не брать, то есть для каждого элемента есть два варианта, всего элементов n, значит число всех вариантов равно 2n, а временная сложность полного перебора равна O(2n).

Эффективность жадного алгоритма нужно считать в двух разных ситуациях: когда ингредиенты уже отсортированы по удельной ценности, и когда неотсортированы. Когда ингредиенты отсортированы, то программа просто пройдет массив ингредиентов и добавит нужные, временная сложность этого процесса O(n), а переборов будет меньше или равно n. Если же массив ингредиентов неотсортирован, то сначала его необходимо отсортировать, это можно сделать за временную сложность O(n\*log(n)) (например, с помощью быстрой сортировки), а затем пройти отсортированный массив за временную сложность O(n) и перебрать не более n вариантов, а значит временная сложность жадного алгоритма при такой ситуации равна O(n\*log(n)).

Таким образом, жадный алгоритм является гораздо менее затратным по времени и числу переборов, нежели полный перебор, но у жадного алгоритма есть недостаток: так как жадный алгоритм находит наиболее эффективное решение лишь на текущем шаге, не задумываясь о будущих выборах, то итоговый результат работы жадного алгоритма может оказаться не самым оптимальным.

# **Вывод**

В результате выполнения работы я:

1. Получил навыки применения методов, позволяющих сократить число переборов в задачах, которые могут быть решены только методом перебора всех возможных вариантов решения.
2. Закрепил знания по тестированию корректности работы программы.
3. Закрепил навыки создания пользовательских интерфейсов.

# **Исходный код программы**

**Файл main.cpp (основной алгоритм программы)**

|  |
| --- |
| #include <iostream>  #include "Task.h"  #include <windows.h>  using namespace std;  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  setlocale(LC\_ALL, "");  cout << "Лабораторная работа №8 \"Алгоритмические стратегии или методы разработки алгоритмов. Перебор и методы его сокращения\". Черномуров Семён\n";  cout << "Выберите номер задания:\n" <<  "1) Задача о рюкзаке (о салате, жире и калориях)\n" <<  "0) Завершить работу программы\n\n";  int choice1;  do {  cin >> choice1;  if (choice1 != 1 && choice1 != 0) cout << "Введено неверное значение, попробуйте снова.\n";  } while (choice1 != 1 && choice1 != 0);  system("cls");  switch (choice1)  {  case 1: {  cout << "Введите размер массивов: ";  int size;  cin >> size;  cout << "\nВведите массив жирности:\n";  int\* fat = new int[size];  for (int i = 0; i < size; i++) {  cin >> fat[i];  }  cout << "\nВведите массив калорий:\n";  int\* calories = new int[size];  for (int i = 0; i < size; i++) {  cin >> calories[i];  }  cout << "\nВведите максимальное число калорий: ";  int maxCalories;  cin >> maxCalories;  BackpackProblem obj(calories, fat, maxCalories, size);  tuple<int, int, vector<pair<int, int>>> answer = obj.solve();  cout << "Набранная жирность: " << get<0>(answer) << "; Набранные калории: " << get<1>(answer)<<".\n";  cout << "Были выбраны ингредиенты (жирность, калорийность, номер в массиве):\n";  for (int i = 0; i < get<2>(answer).size(); i++) {  cout << get<2>(answer)[i].first << " " << get<2>(answer)[i].second << " ("<<i+1<<")\n";  }  break;  }  case 0:  return 0;  }  cout << "\n\n";  main();  } |

**Файл Task.h (заголовочный файл класса BackpackProblem)**

|  |
| --- |
| #pragma once  #ifndef TASK\_H  #define TASK\_H  #include <utility>  #include <tuple>  #include <vector>  class BackpackProblem {  private:  int\* calories;  int\* fat;  int maxCalories;  int size;  public:  BackpackProblem(int\* calories, int\* fat, int maxCalories, int size);  std::tuple<int, int, std::vector<std::pair<int, int>>> solve();  };  #endif // !TASK\_H |

**Файл Task.cpp (реализация методов класса BackpackProblem)**

|  |
| --- |
| #include "Task.h"  #include <algorithm>  #include <iostream>  using namespace std;  //Конструктор класса BackpackProblem  BackpackProblem::BackpackProblem(int\* calories, int\* fat, int maxCalories, int size) {  this->calories = calories;  this->fat = fat;  this->maxCalories = maxCalories;  this->size = size;  }  //Метод решения задачи  tuple<int, int, vector<pair<int,int>>> BackpackProblem::solve() {  pair<int, int>\* arr = new pair<int, int>[size];  for (int i = 0; i < size; i++) {  arr[i].first = fat[i];  arr[i].second = calories[i];  }  sort(arr, arr + size, [](pair<int, int> a, pair<int, int> b) {return a.first \* 1.0 / a.second > b.first \* 1.0 / b.second; });  int currentCalories = 0;  int currentFat = 0;  vector<pair<int, int>> values;  for (int i = 0; i < size; i++) {  if (currentCalories + arr[i].second <= maxCalories) {  currentCalories += arr[i].second;  currentFat += arr[i].first;  values.push\_back(make\_pair(arr[i].first, arr[i].second));  }  }  delete[] arr;  //cout << currentCalories << " " << currentFat;  return make\_tuple(currentFat, currentCalories, values);  } |