**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Информационных систем**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Типы данных и их внутреннее представление в памяти

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Кольцов К.Э |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

**Цель работы.**

Разработать алгоритм и написать программу, которая позволяет:

1) Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

2) Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.

3) Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

**Основные теоретические положения.**

Числа в памяти компьютера представлены в виде целого двоичного числа. Для знаковых чисел старший разряд числа используется для обозначения знака: ноль для неотрицательных чисел и единица для отрицательных. Для простоты внутренней арифметики последние представлены в дополненном коде. Он получается прибавлением единицы к обратному коду числа (в котором все разряды модуля числа инвертированы).

Вещественные типы бывают только знаковыми. Для них отличается представление в памяти. Они переводятся в нормализованную форму двоичной системы счисления. Так как нормализованная мантисса в двоичной системе счисления больше либо равна единице и меньше двух, значит она всегда начинается с единицы и её можно не записывать в память, освободив место для еще одного разряда порядка или мантиссы. У порядка знакового бита нет, вместо этого к нему прибавляется половина диапазона. Например, порядок равный нулю будет начинаться нулём, а все остальные разряды будут единицами.

Для нахождения размера занимаемой памяти используется функция sizeof, возвращающая количество байт для соответствующего типа данных. Для вывода представления чисел в памяти используется & (оператор побитового И) вместе с числом с единственной единицей в двоичной записи (два в неотрицательной степени), а для вещественных типов данных union (объединение), для получения доступа в области памяти через целочисленные типы (int для float и long long для double).

**Экспериментальные результаты.**

Программа выводит на экран, сколько памяти (в байтах) на компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.

Затем по порядку считываются, обрабатываются и выводятся двоичные представления в памяти типов int, float, double.

После этого программа завершает работу (см. рис. 1)

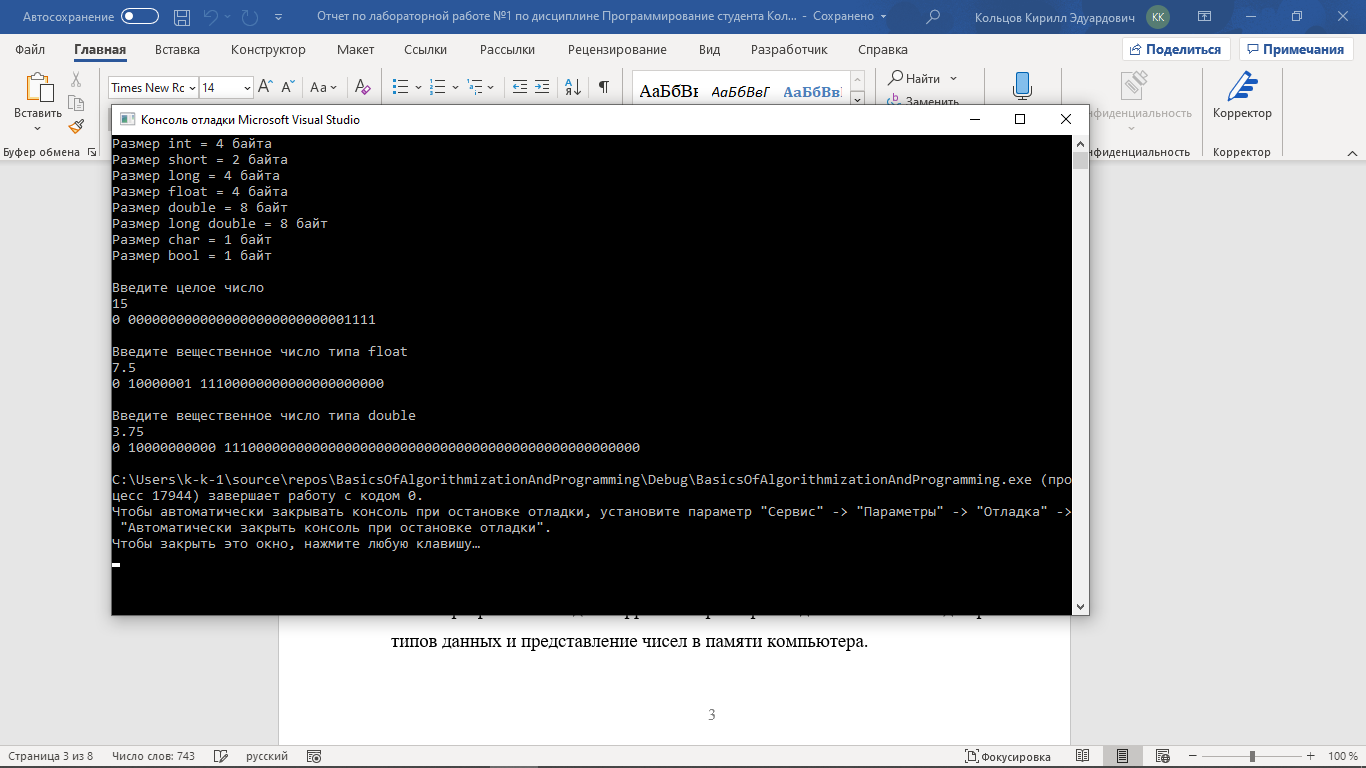


Рисунок 1 – Пример работы программы

**Обработка результатов эксперимента.**

Программа выводит корректные размеры выделяемой памяти для разных типов данных и представление чисел в памяти компьютера.

**Выводы.**

В ходе данной лабораторной работы я изучил различные способы сортировки массивов и работы с ними.

ПРИЛОЖЕНИЕ А

Листинг программы

#include <iostream>

#include <ctime>

#include <chrono>

#define N 100

bool wasArrSort;

void newArr(int\*, int\*);

void quicksort(int\*, int, int);

int findElem(int\*, int, int, int);

int simpFindElem(int\*, int, int, int);

void findAllMed(int\*, int, int);

double findPlace(int\*, int, int, int);

void userFindLess(int\*);

void userFindMore(int\*);

void userSwap(int\*);

void userFindNumber(int\*);

int main(){

std::cout.setf(std::ios::fixed);

std::cout.precision(7);

std::srand(time(0));

setlocale(LC\_ALL, "Russian");

int arrUnsort[N], arrSort[N];

newArr(arrUnsort, arrSort);

char answer;

bool flag = true;

while (flag) {

std::cout << "Введите команду для работы с массивом\n";

std::cin >> answer;

switch (answer)

{

case 'h':

std::cout << "h: список команд\n" \

"s: заново создать массив\n" \

"b: найти все элементы больше вашего числа\n" \

"l: найти все элементы меньше вашего числа\n" \

"m: найти индексы среднего элемента\n" \

"w: поменять местам пару элементов в массиве\n" \

"f: найти индекс элемента, равного вашему числу\n" \

"e: завершить работу программы";

break;

case 's':

newArr(arrUnsort, arrSort);

break;

case 'b':

userFindMore(arrSort);

break;

case 'l':

userFindLess(arrSort);

break;

case 'm':

findAllMed(arrSort, 0, N - 1);

break;

case 'w':

userSwap(arrSort);

break;

case 'f':

userFindNumber(arrSort);

break;

case'e':

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

system("pause");

return 0;

}

void newArr(int\* arrUnsort, int\* arrSort) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> sortTime, findTime;

int min, max;

std::cout << "Несортированный массив: ";

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

{

arrUnsort[i] = (std::rand() % 199) - 99;

arrSort[i] = arrUnsort[i];

std::cout << arrUnsort[i] << " ";

}

wasArrSort = false;

std::cout << "\n\n";

start = std::chrono::system\_clock::now();

quicksort(arrSort, 0, N - 1);

end = std::chrono::system\_clock::now();

sortTime = end - start;

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arrSort[i] << ' ';

std::cout << "\n\n";

std::cout << "Сортировка заняла " << sortTime.count() << " сек.\n\n";

start = std::chrono::system\_clock::now();

min = max = arrUnsort[0];

for (int i = 1;i < N;i++) {

if (arrUnsort[i] > max)

max = arrUnsort[i];

else if (arrUnsort[i] < min)

min = arrUnsort[i];

}

end = std::chrono::system\_clock::now();

std::cout << "Минимальный элемент = " << min << ", максимальный элемент = " << max << '\n';

findTime = end - start;

std::cout << "Поиск в неотсортированном массиве занял " << findTime.count() << " сек.\n";

start = std::chrono::system\_clock::now();

min = arrSort[0];

max = arrSort[N - 1];

end = std::chrono::system\_clock::now();

findTime = end - start;

std::cout << "Поиск в отсортированном массиве занял " << findTime.count() << " сек.\n";

std::cout << "Сортировка и поиск в массиве заняли " << (findTime + sortTime).count() << " сек.\n\n";

}

void quicksort(int\* arr, int begin, int end)

{

int f = begin, l = end, mid = arr[(f + l) / 2];

while (f <= l)

{

while (arr[f] < mid) f++;

while (arr[l] > mid) l--;

if (f <= l)

{

std::swap(arr[f], arr[l]);

f++;

l--;

}

}

if (begin < l) quicksort(arr, begin, l);

if (f < end) quicksort(arr, f, end);

}

void findAllMed(int\* arr, int begin, int end) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int med = (arr[begin] + arr[end]) / 2;

std::cout << "Среднее значение = " << med << '\n';

int sum = 0, count = findElem(arr, begin, end, med);

if (count == -1) {

std::cout << "Средний элемент = " << med << " в массиве не найден\n\n";

}

else {

while (arr[count] == med)

count--;

count++;

std::cout << "Средний элемент = " << med << " и находится по индексу ";

while (arr[count] == med) {

std::cout << count << ", ";

count++;

sum++;

}

std::cout << "и всего их " << sum << "\n\n";

}

}

int findElem(int\* arr, int begin, int end, int elem) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

if ((elem < arr[begin]) || (elem > arr[end]))

return -1;

double count = findPlace(arr, begin, end, elem);

if (((int)count) == ((int)(count + 0.5)))

return (int)count;

return -1;

}

int simpFindElem(int\* arr, int begin, int end, int elem) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

if ((elem < arr[begin]) || (elem > arr[end]))

return -1;

for (int i = begin;i <= end;i++)

if (arr[i] == elem)

return i;

return -1;

}

double findPlace(int\* arr, int begin, int end, int elem) {

if ((end - begin) < 2) {

if (elem == arr[end])

return end;

if (elem == arr[begin])

return begin;

return (double)(end + begin) / 2;

}

else{

int count = (begin + end) / 2;

if (elem > arr[count])

return findPlace(arr, count, end, elem);

return findPlace(arr, begin, count, elem);

}

}

void userFindLess(int\* arr) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int ab;

double placeAB;

std::cout << "Введите число a, чтобы узнать, сколько чисел меньше него\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

if (ab < arr[0])

std::cout << "0 элементов меньше чем " << ab << "\n\n";

else {

int count;

if (ab > arr[N - 1])

std::cout << "100 элементов меньше чем " << ab << "\n\n";

else {

double countD = findPlace(arr, 0, N - 1, ab);

if (((int)countD) == ((int)(countD + 0.5)))

count = (int)countD;

else

count = (int)(countD - 0.5);

while (arr[count] == ab)

count--;

std::cout << count + 1<< " элементов меньше чем " << ab << "\n\n";

}

}

}

void userFindMore(int\* arr) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int ab;

double placeAB;

std::cout << "Введите число b, чтобы узнать, сколько чисел больше него\n";

while (true) {

std::cin >> ab;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

if (ab > arr[N - 1])

std::cout << "0 элементов больше чем " << ab << "\n\n";

else {

int count;

if (ab < arr[0])

std::cout << "100 элементов больше чем " << ab << "\n\n";

else {

double countD = findPlace(arr, 0, N - 1, ab);

if (((int)countD) == ((int)(countD + 0.5)))

count = (int)countD;

else

count = (int)(countD + 0.5);

while (arr[count] == ab)

count++;

std::cout << N - count << " элементов больше чем " << ab << "\n\n";

}

}

}

void userFindNumber(int\* arr) {

if (!wasArrSort) {

std::cout << "Для выполнения операции, нужно отсортировать массив\n";

wasArrSort = true;

std::cout << "Отсортированный массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}

int a;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> binFindTime, simpFindTime;

std::cout << "Введите число, которое нужно найти в массиве\n";

while (true) {

std::cin >> a;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

int countBin, countSimp;

start = std::chrono::system\_clock::now();

countBin = findElem(arr, 0, N - 1, a);

end = std::chrono::system\_clock::now();

binFindTime = end - start;

start = std::chrono::system\_clock::now();

countSimp = simpFindElem(arr, 0, N - 1, a);

end = std::chrono::system\_clock::now();

simpFindTime = end - start;

if (countBin == -1)

std::cout << "Вашего числа нет в массиве,\n";

else

std::cout << "Ваше число соответствует элементу массива с индексом " << countBin << " и " << countSimp << '\n';

std::cout << "На поиск ответа ушло " << binFindTime.count() << " сек. бинарным поиском и " << simpFindTime.count() << " сек. простым перебором\n\n";

}

void userSwap(int\* arr) {

int a, b;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> swapTime;

std::cout << "Перестановка элементов массива. Введите индексы через пробел\n";

while (true) {

std::cin >> a >> b;

if (std::cin)

break;

std::cout << "Вы неправильно ввели пару чисел. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

if (a == b)

std::cout << "Вы ввели одинаковые числа\n";

else

wasArrSort = false;

start = std::chrono::system\_clock::now();

std::swap(arr[a], arr[b]);

end = std::chrono::system\_clock::now();

swapTime = end - start;

std::cout << "Перестановка элементов заняла " << swapTime.count() << " сек.\n";

std::cout << "Измененый массив: ";

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

std::cout << arr[i] << ' ';

std::cout << "\n\n";

}