1. **Введение**

Microsoft Visual Studio — это программная среда по разработке приложений для ОС Windows, как консольных, так и с графическим интерфейсом.

Функциональная структура среды включает в себя:

1) редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинг кода;

2) отладчик кода;

3) редактор форм, предназначенный для упрощённого конструирования графических интерфейсов;

4) веб-редактор;

5) дизайнер классов;

6) дизайнер схем баз данных.

Visual Studio также позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов (для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментов для прочих аспектов процесса разработки программного обеспечения).

В комплект входят следующие основные компоненты:

1. Visual Basic.NET – для разработки приложений на VisualBasic;
2. Visual C++ – на традиционном языке C++;
3. Visual C# – на языке C# (Microsoft);
4. Visual F# – на F# (Microsoft Developer Division).

Интегрированная среда разработки (IntegratedDevelopmentEnvironment – IDE) Visual Studio предлагает ряд высокоуровневых функциональных возможностей, которые выходят за рамки базового управления кодом.

Ниже перечислены основные преимущества IDE-среды Visual Studio:

1. Встроенный Web-сервер;
2. Поддержка множества языков при разработке;
3. Интуитивный стиль кодирования;
4. Более высокая скорость разработки;
5. Возможности отладки.

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

Своими корнями он уходит в язык Си, который был разработан в 1969—1973 годах в компании Bell Labs программистом Деннисом Ритчи. В начале 1980-х годов датский программист Бьерн Страуструп, который в то время работал в компании Bell Labs, разработал С++ как расширение к языку Си. Фактически вначале C++ просто дополнял язык Си некоторыми возможностями объектно-ориентированного программирования. И поэтому сам Страуструп вначале называл его как "C with classes" ("Си с классами").

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где С++ тоже нашел свое применение. И даже в веб-разработке также можно использовать С++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем С++ – язык широкого пользования, на котором можно создавать практически любые виды программ.

1. **Разработка и описание алгоритма**

Сортировка выбором – один из самых известных методов сортировки массива. Алгоритм был разработан в 1950 году и отличается простотой и легкостью в реализации.

Общая идея алгоритма состоит в следующем:

* Находим номер минимального значения в текущем списке.
* Производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции).
* Теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы.

**Достоинства алгоритма сортировки выбором:**

* Прост в реализации.
* Эффективен для небольших массивов.
* Не требует дополнительной памяти для сортировки.

**Недостатки алгоритма:**

* Неэффективен для больших списков из-за квадратичной скорости.
* Неустойчив.
* При худшем, среднем и лучшем случае время остается постоянным.

1. **Описание программы**

В программе для сортировки выбором подключены следующие заголовочные файлы: *iostream* – заголовочный файл с классами, функциями и переменными для организации ввода-вывода; ***fstream* – заголовочный файл,** предоставляющий функционал для считывания данных из файла и для записи в файл; *Using namespace std* – использование стандартного пространства имен.

#include <iostream>

#include <fstream>

using namespace std;

Далее идет текст функции choice\_sort. В первом цикле выбирается крайний левый элемент неотсортированного массива. Далее во вложенном цикле мы сравниваем элементы массива с крайним левым элементом неотсортированного массива. Если сравниваемый элемент массива меньше чем минимальный найденный элемент массива, то найденный элемент становится минимальным, а индекс запоминается. После вложенного цикла мы сравниваем минимальный индекс с индексом крайнего правого элемента. Если они не равны, то меняем местами элемент массива с минимальным индексом с крайним правым элементом неотсортированного массива.

void choice\_sort(int n, int mass[]) {

int buf = 0;

for (int i = 0; i < n - 1; i++) {

int min\_i = i;

for (int j = i + 1; j < n; j++) {

if (mass[j] < mass[min\_i]) { min\_i = j; }

}

if (min\_i != i) {

buf = mass[i];

mass[i] = mass[min\_i];

mass[min\_i] = buf;

}

}

}

Далее располагается текст функции создания и заполнение неотсортированного массива. Вначале создается файл, далее запускается цикл, который записывает случайные числа в файл. Далее файл закрывается.

void create\_file(int n) {

ofstream fout("no\_sort\_array.txt");

srand(time(0));

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

fout << rand() % 20001 - 10000 << endl;

}

fout.close();

}

Далее идет текст программы меню. Вначале выводим пункты меню. Потом запускаем бесконечный цикл с предусловием и запрашиваем в нем число, которое является пунктом меню.

void menu() {

int n, count;

string name, t = ".txt";

cout << "\n\t1. Случайный ввод\n\t2. Ручной ввод\n\t3. Ввод с файла\n\t4. Выход\n\n";

while (true){

cout << "\tВыберите пункт: ";

cin >> n;

cout << endl;

Далее используем конструкцию switch case. Выбор происходит от номера пункта. В первом случае мы сначала запрашиваем количество элементов массива и создаем массив. Далее вызываем функцию создания файла create\_file. Затем с помощью цикла считываем числа из файла в массив. Потом запускается функция сортировки массива с замером времени сортировки. Далее выводим сообщение о времени сортировки массива, а после с помощью цикла массив записывается в новый файл.

switch (n) {

case 1: {

int n, buf = 0;

cout << "\tВведите количество элементов: ";

cin >> n;

int\* mass = new int[n];

create\_file(n);

ifstream fin("no\_sort\_array.txt");

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

fin >> mass[i];

}

fin.close();

time\_t start = clock();

choice\_sort(n, mass);

time\_t stop = clock();

double time = (stop - start) / 1000.0;

cout << "\n\tВремя сортировки массива состовляет: " << time << " секунд\n";

ofstream fout("sort\_array.txt");

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

fout << mass[i] << endl;

}

fout.close();

exit(0);

}

Во втором случае мы сначала запрашиваем количество элементов массива и создаем массив. Далее поэлементно вводим массив с клавиатуры. Затем запускается функция сортировки массива с замером времени сортировки. Далее выводим сообщение о времени сортировки массива, а после с помощью цикла выводим отсортированный массив на экран.

case 2: {

int n;

cout << "\tВведите количество элементов: ";

cin >> n;

cout << endl;

int\* mass = new int[n];

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

cout << "\tВведите " << i + 1 << " элемент массива: ";

cin >> mass[i];

cout << endl;

}

time\_t start = clock();

choice\_sort(n, mass);

time\_t stop = clock();

double time = stop - start;

cout << "\tВремя сортировки массива состовляет: " << time << " милисекунд\n\n" << "Сортированный массив: ";

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

cout << mass[i] << " ";

}

exit(0);

}

В третьем случае мы сначала запрашиваем имя заранее подготовленного файла. В файле сначала идет количество элементов, а далее уже сами числа. Далее считываем первый элемент, который является количеством элементов. Затем создаем массив из считанного количества элементов. После с помощью цикла считываем числа из файла в массив. Потом запускается функция сортировки массива с замером времени сортировки. Далее выводим сообщение о времени сортировки массива, а после с помощью цикла массив записывается в новый файл.

case 3: {

cout << "\tВведите имя файла: ";

cin >> name;

name += t;

ifstream fin(name);

fin >> count;

int\* mass = new int[count];

for (int i = 0; i < count; i++) {

fin >> mass[i];

}

fin.close();

time\_t start = clock();

choice\_sort(count, mass);

time\_t stop = clock();

double time = (stop - start) / 1000.0;

cout << "\n\tВремя сортировки массива состовляет: " << time << " секунд";

ofstream fout("sort\_array.txt");

for (int i = 0; i < count; i++) {

fout << mass[i] << endl;

}

fout.close();

exit(0);

}

В четвёртом случае происходит завершение программы, а в остальных случаях выводится сообщение об ошибке и цикл начинается заново.

case 4: { exit(0); }

default: { cout << "Такого пункта нет, попробуйте ещё раз" << endl;}

}

Затем идет текст главной программы. Сначала выполняется русификация, а после запускается функция меню.

int main(){

setlocale(0, "");

menu();

return 0;

}

1. **Схема программы**
   1. **Блок-схема программы**

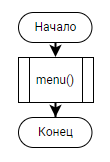


Рисунок 1 – Блок-схема программы

* 1. **Блок-схема алгоритма сортировки**

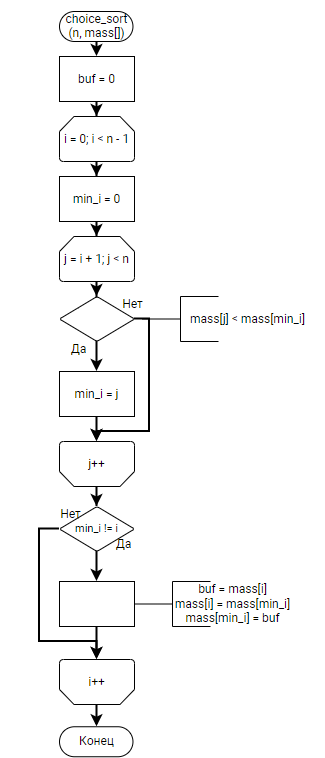
****

Рисунок 2 – блок-схема алгоритма сортировки

* 1. **Блок-схема функции создания файла**

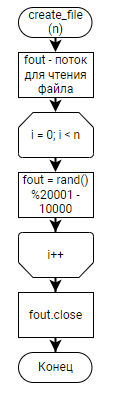
****

Рисунок 3 – блок-схема функции создания файла

* 1. **Блок-схема функции меню**

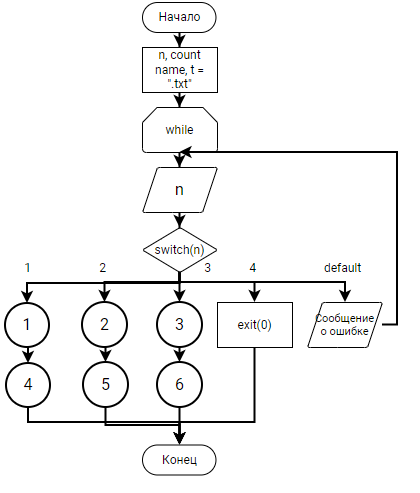


Рисунок 4 – часть блок-схемы функции меню

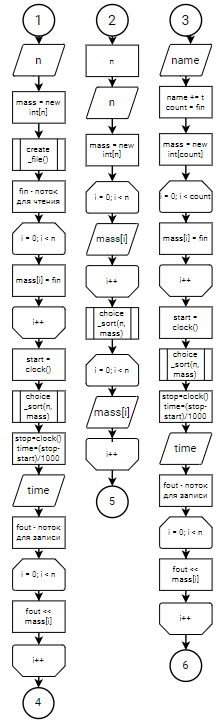


Рисунок 5 – оставшаяся часть блок-схемы функции меню

1. **Тестирование программы**
   1. **Тестирование на разных наборах данных**

Тестовый набор данных представлен в таблице 1. Результаты тестирования приведены в Приложении А на рисунках А.1 – А.11.

Таблица 1.

|  |  |  |
| --- | --- | --- |
|  | Размер массива size | Время выполнения сортировки в секундах |
| 1 | 10000 |  |
| 2 | 20000 |  |
| 3 | 30000 |  |
| 4 | 40000 |  |
| 5 | 50000 |  |
| 6 | 60000 |  |
| 7 | 70000 |  |
| 8 | 80000 |  |
| 9 | 90000 |  |
| 10 | 100000 |  |
| 11 | 110000 |  |

* 1. **Анализ полученных результатов тестирования**

1. **Отладка**

В качестве среды разработки была выбрана программа Microsoft Visual Studio, которая содержит в себе все необходимые средства для разработки и отладки модулей и программ.

Для отладки программы использовались точки остановки и пошаговое выполнение кода программы, анализ содержимого локальных переменных.

Точки останова – это прерывание выполнения программы, при котором выполняется вызов отладчика. Отладчик является инструментом для поиска и устранения ошибок в программе, с помощью которого можно исследовать состояние программы.

Команда шаг с заходом (step into) выполняет следующую инструкцию в обычном пути выполнения программы, а затем приостанавливает выполнение программы, чтобы мы могли проверить состояние программы с помощью отладчика. Если выполняемый оператор содержит вызов функции, шаг с заходом заставляет программу перескакивать в начало вызываемой функции, где она приостанавливается.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. После завершения написания программы, мною были выявлены и исправлены ошибки.

1. **Совместная разработка**

Для удобства был создан репозиторий. Ссылка на него: <https://github.com/FuneTome/praktika.git>

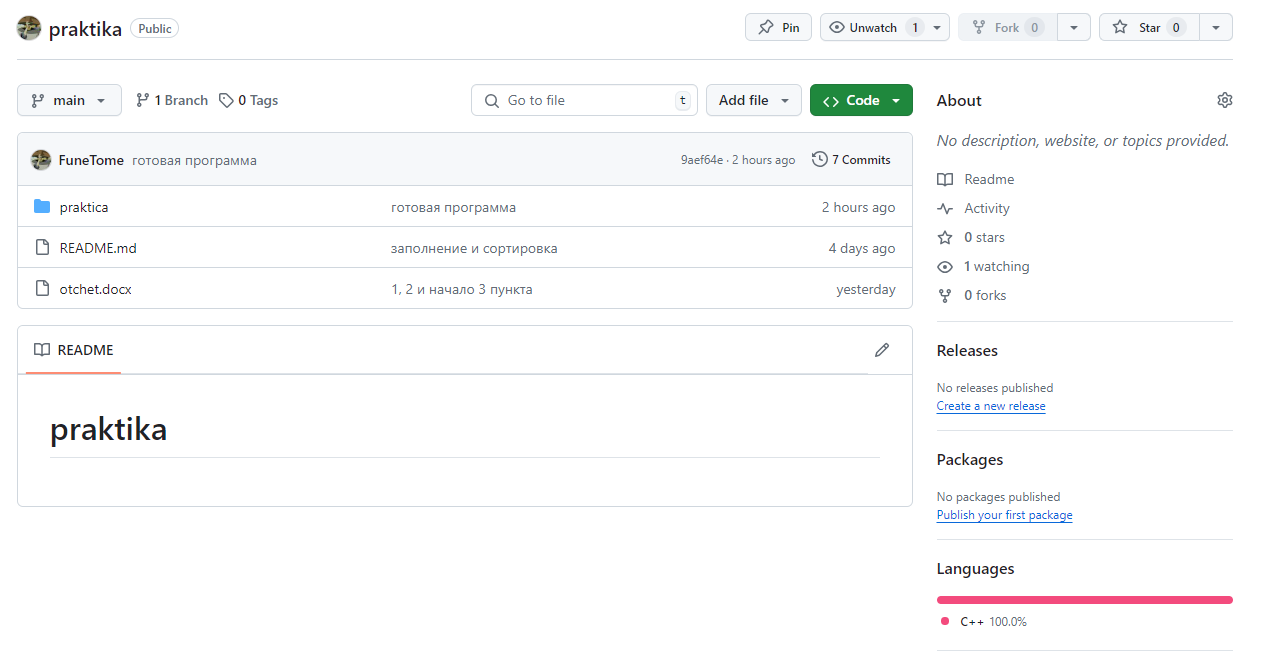


Рисунок 6 – репозиторий и ветка main

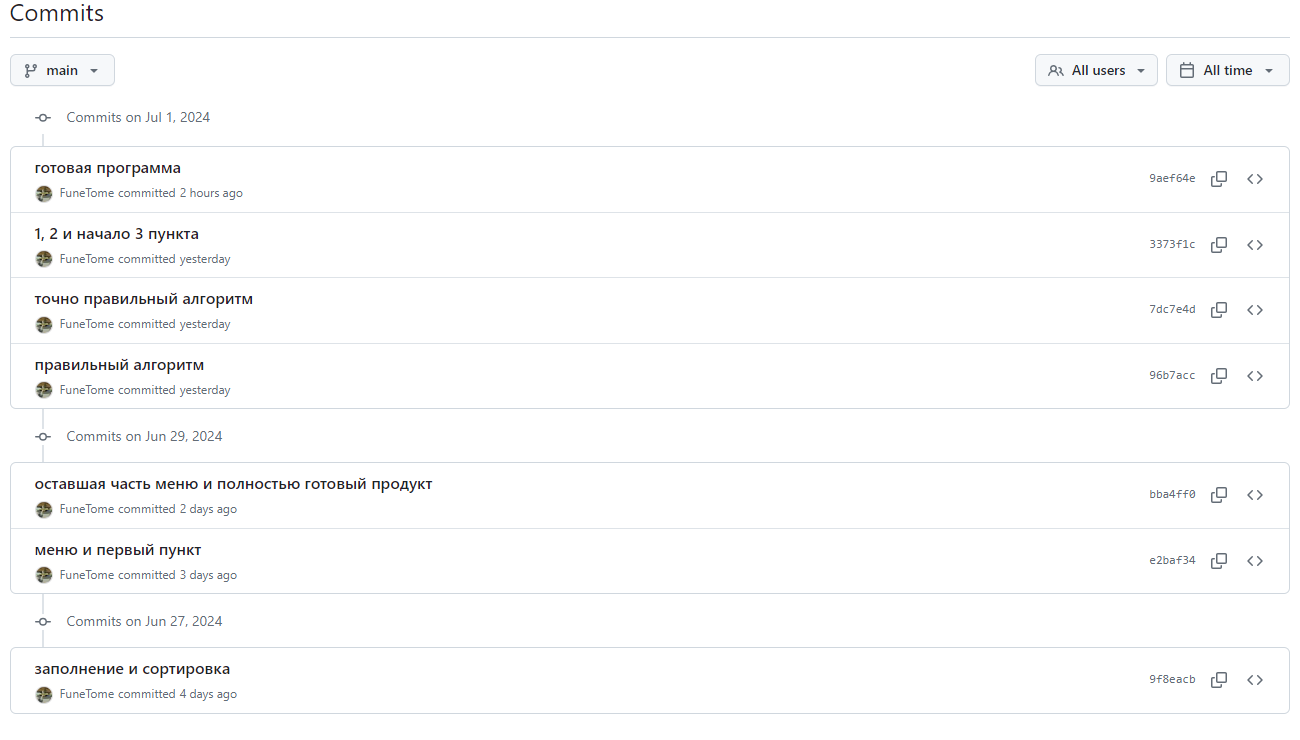


Рисунок 7 – созданные коммиты

Для загрузки на локальный и глобальный репозиторий были использованы основные команды git bash.