Функция main() проверяет введенные в консоль флаги --toFile и --fromFile независимо от их порядка. В случае отсутствия данных флагов текст следует набрать в консоли, результат сортировки так же будет выведен в консоль. Для завершения набора текста необходимо ввести "exit".

```
int main(int argc, char** argv)
{
    bool fromF, toF = false;
    char* fromFName = new char[1024];
    char* toFName = new char[1024];
    if(argc > 1)
    {
        if (strcmp(argv[1], "--fromFile") == 0)
        {
            if (argc == 2) {
                cout << "Wrong input" << endl;
                return 1;
            }
            fromF = true;
            fromFName = argv[2];
        }
        else if (strcmp(argv[1], "--toFile") == 0)
        {
            if (argc == 2) {
                cout << "Wrong input" << endl;
                return 1;
            }
            toF = true;
            toFName = argv[2];
        }
        if(argc > 3)
        {
            if (strcmp(argv[3], "--fromFile") == 0) {
                if (argc == 4)
                {
                    cout << "Wrong input" << endl;
                    return 1;
                }
                if (fromF)
                {
                    cout << "Wrong input" << endl;
                    return 1;
                }
                fromF = true;
                fromFName = argv[4];
            }
            if (strcmp(argv[3], "--toFile") == 0)
            {
                if (argc == 4)
                {
                    cout << "Wrong input" << endl;
                    return 1;
                }
                if (toF)
                {
```

```
                    cout << "Wrong input" << endl;
                    return 1;
                }
                toF = true;
                toFName = argv[4];
            }

        }
    }
    if (!fromF)
    {
        ofstream f1;
        f1.open("fileTemp");
        char* s = new char[1024];
        if (!f1.is_open())
        cout << "Write exit to stop" << endl;
        while (true)
        {
            cin >> s;
            if (strcmp(s,"exit") == 0)
            {
                break;
            }
        f1 << s << endl;
        }
        f1.close();
        strcpy(fromFName, "fileTemp");

    }
    processing(fromFName, toF, toFName);

    return 0;
}
```

processing читает текст из предоставленного файла и выводит отсортированные в порядке неубывания строки текста и также выводит в обратном порядке самое длинное предложение теста в файл NINE.txt.

```
void processing(char* fromFName, bool toF, char* toFName)
{
    ifstream f;
    f.open(fromFName);
    int n = 1000;
    int k = 1000;
    char ch;
    char* s = new char[n];
    s[0] = '\0';
    char** arr = new char*[k];
    arr[0] = NULL;
    while(f.get(ch))
    {
        if (ch == '\n')
            ch = ' ';
        s = addCh(s, ch, &n);
        if (ch == '.')
```

```
        {
            arr = addStr(arr, s, &k);
            n = 1000;
            s = new char[n];
            s[0] = '\0';
        }
    }
    for (int i = 0; i < len(arr); i++)
    {
        if (arr[i][0] == ' ')
            arr[i] ++;
    }
    sort(arr);
    char* longest = revLongest(arr[len(arr)-1]);
    ofstream nine("NINE.txt");
    nine << longest;
    nine.close();
    if (toF)
        writing(arr, toFName);
    else
        print(arr);

}

len возвращает длину массива масиввов.
int len(char** arr)
{
    int i = 0;
    while (arr[i] != NULL)
        i++;
    return i;
}

sort сортирует предложения пузырьком.
void sort(char** arr)
{
    for (int i = 0; i < len(arr); i++)
    {
        for(int j = 0; j < len(arr)-1; j++) {
            if (strlen(arr[j]) > strlen(arr[j+1]))
            {
                char* t = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = t;
            }
        }
    }

}
addCh добавляет к строке прочитанный символ и выделяет больше памяти для
символа в случае недостатка.
char* addCh(char* s, char ch, int* n) {
    if (strlen(s) == *n - 1) {
        *n += 100;
```

```
        char* snew = new char[*n];
        strcpy(snew, s);
        int x = strlen(s);
        delete s;
        snew[x] = ch;
        snew[x+1] = '\0';
        return snew;
    }
    int x = strlen(s);
    s[x] = ch;
    s[x+1] = '\0';
    return s;
}
```

addStr добавляет к массиву строк строку и выделяет больше памяти для строки в случае недостатка.

```
char** addStr(char** arr, char* s, int* k)
{
    if (len(arr) == *k - 1)
    {
        *k += 100;
        char** arrnew = new char*[*k];
        int i = 0;
        while (arr[i] != NULL)
        {
            arrnew[i] = arr[i];
            i++;
        }
        arrnew[i] = NULL;
        delete arr;
        arr = NULL;
        int x = len(arr);
        arrnew[x] = s;
        arrnew[x+1] = NULL;
        return arrnew;
    }
    int x = len(arr);
    arr[x] = s;
    arr[x+1] = NULL;
    return arr;
}
```

print выводит предложения в терминале.

```
void print(char** arr)
{
    for(int i = 0; i < len(arr); i++)
    {
        cout << arr[i] << endl;
    }
}
```

writing записывает предложения в файл.

```
void writing(char** arr, char* toFName)
{
```

```
    ofstream f3(toFName);
    for(int i = 0; i < len(arr); i++)
    {
        f3 << arr[i];
        if (i != len(arr)-1)
            f3 << endl;
    }
    f3.close();
}
```

revLongest и revAddStr переварачивают самое длинное предложение, не перевроячивая слова.

```
char* revAddStr(char* s1, char* s2, int n)
{
    int a, b;
    a = strlen(s1);
    b = strlen(s2);
    char* sre = new char[a + b + 2];
    strncpy(sre, s2, n);
    sre[n] = '\0';
    strcat(sre, " ");
    sre[n+1] = '\0';
    strcat(sre, s1);
    sre[a + b + 1] = '\0';
    return sre;
}


char* revLongest(char* strTemp)
{
    char* revFin = new char[strlen(strTemp) + 1];
    revFin[0] = '\0';
    int i = 0;
    while (i < strlen(strTemp))
    {
        if (strTemp[i] != ' ')
        {
            i++;
            continue;
        }
        revFin = revAddStr(revFin, strTemp, i);
        strTemp += i + 1;
        i = 0;
    }
    revFin = revAddStr(revFin, strTemp, strlen(strTemp));
    return revFin;
}
```