1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт Кибербезопасности и Защиты Информации**

**КУРСОВАЯ РАБОТА**

1. на тему «**Оптимизация алгоритма решения крипторифмов**»
2. по дисциплине «Языки программирования»
3. Выполнили
4. студент гр. 4851003/10001 Цыганков И.С
5. <*подпись*>

Руководитель

1. ст. преподаватель Семьянов П.В.

<*подпись*>

Санкт-Петербург

2023

ВВЕДЕНИЕ

Оптимизация кода - различные методы преобразования кода ради улучшения его характеристик и повышения эффективности. Среди ее целей можно указать уменьшения объема кода, объема используемой программой оперативной памяти, ускорение работы программы, уменьшение количества операций ввода вывода. Существуют различные оптимизации: алгоритмические, машинно-независимые, оптимизации переходов, машинно-зависимые, оптимизации на уровне ассемблера и другие.

Из анализа исследуемой литературы можно сделать вывод, что все компиляторы с языка Си предназначены для генерации наиболее быстрых и компактных программ и содержат в себе возможность оптимизации. Это еще раз подтверждает важную роль оптимизации кода в программировании.

Объектом исследования является неоптимизированный программный код.

Предметом исследования является алгоритм, используемый для решения крипторифмов.

ОПТИМИЗАЦИИ НА УРОВНЕ АЛГОРИТМА

Ранее решение крипторифма получалось перебором всех чисел для каждого символа из трёх слов. Алгоритм не является оптимальным так как можно существенно уменьшить количество перебираемых значений.

В модифицированном алгоритме сразу после ввода слов проверяются два случая не требующие подбора:

1. Длинна третьего слова (результата сложения) больше длинны

суммируемых слов. Тогда в высшем разряде результата сложения стоит единица (пример такого случая представлен ниже).

send send m = 1

+ more => + more

money 1oney

Так как в высшем разряде происходит операция 0 + 0 = m, а слово не может начинаться с 0 => был переход через разряд и m = 1.

1. Длинна третьего слова (результата сложения) больше длинны

суммируемых слов и длинна одного из суммируемых слов больше второго. Тогда в высшем разряде результата сложения стоит единица, в высшем разряде большего (по длине) из суммируемых слов стоит 9, а во втором разряде с лева результата сложения 0(пример такого случая представлен ниже).

attractions 9ttractions a = 9

+ intentions => + intentions e = 0

regeneration 10generation r = 1

Так как в высшем разряде происходит операция 0 + 0 = r, а слово не может начинаться с 0 => был переход через разряд и r = 1. В втором справа разряде происходит операция a + 0 = e, при этом, так как нужен переход через разряд, результат сложения ≥ 10, так как a ≤ 9 => был переход через разряд, a = 9, результат сложения = 10, e = 0.

Подбор значений осуществляется блоками из трех символов, стоящих в одном разряде начиная с конца слов. Для этого все три слова выравниваются под длину результата сложения (у слов длинна которых меньше результата сложения в начале добавляются пробелы). Подбор решения крипторифма осуществляется рекурсивно в функции permutation, в функции для каждого символа в блоке проверяется присвоено ли символу какое-то число. Для блока рассматриваются следующие случаи (обозначим w1 – число, соответствующее символу из первого суммируемого слова, w2 – число, соответствующее символу из второго суммируемого слова, w3 – число, соответствующее символу из результата сложения, c – значение перехода через разряд):

1. Всем символам блока присвоено значение. Проверяем корректность

сложения (w3 == w1 + w2 + c), если результат верный переходит к разряду выше, иначе возвращается на предыдущий.

1. Присвоено значение результату сложения и одному из суммируемых

символов (w1 или w2), тогда неизвестный символ = w3 – известный символ – с или w3 – известный символ – с + 10. Проверяем занято ли полученное число и переходим к разряду выше или ниже в зависимости от результата проверки.

1. Присвоено значение w3, w1 и w2 – неизвестны.
   1. Символ первого слова == символу из второго слова. Если с = 0, то

w1 = w2 = w3/2 или w1 = w2 = (w3+10)/2. Если с = 1, то w1 = w2 = (w3-1)/2 или w1 = w2 = (w3-1+10)/2. Проверяем занято ли полученное число и переходим к разряду выше или ниже в зависимости от результата проверки.

* 1. Символ первого слова != символу из второго слова. Циклически

подбираем значение w1 от 1 до 9, w2 = w3 – w1 – c или w2 = w3 + 10 – w1 – c. Проверяем заняты ли полученные числа и переходим к разряду выше или ниже в зависимости от результата проверки.

1. Присвоено значение w1, w2, тогда w3 = (w2 + w1 + c)%10. Проверяем

занято ли полученное число и переходим к разряду выше или ниже в зависимости от результата проверки.

1. Присвоено значение одному из суммируемых символов (w1 или w2),

циклически подбираем значение неизвестного символа от 1 до 9, w3 = (w2 + w1 + c)%10. Проверяем заняты ли полученные числа и переходим к разряду выше или ниже в зависимости от результата проверки.

1. Всем трём символам не присвоено значение.
   1. Символы во всех трёх словах одинаковы. Если с = 0, то w1 = w2 =

w3 = 0. Если с = 1, то w1 = w2 = w3 = 9 (9 + 9 + 1 = 19). Проверяем занято ли полученное число и переходим к разряду выше или ниже в зависимости от результата проверки.

* 1. Символ в одном из суммируемых слов равен символу в результате

(a + b = a). Если с = 0, то уникальный символ = 0 (b = 0), а второй символ принимает любое значение от 1 до 9, циклически подбираем его. Если с = 1, то уникальный символ = 9, а второй символ принимает любое значение от 1 до 9, циклически подбираем его. Проверяем занято ли полученное число и переходим к разряду выше или ниже в зависимости от результата проверки.

* 1. Символы суммируемых слов равны друг другу и не равны

результату. Если с = 0, циклически подбираем результат сложения от 2 до 18 с шагом 2, w3 = результат сложения%10 w1 = w2 = результат сложения/2. Если с = 1, циклически подбираем результат сложения от 1 до 19 с шагом 2, w3 = результат сложения%10 w1 = w2 = результат сложения/2. Проверяем заняты ли полученные числа и переходим к разряду выше или ниже в зависимости от результата проверки.

* 1. Если символы в блоке не повторяются, то циклически подбираем

значения w1 и w2, таки что w1 != w2, w3 = (w1 + w2)%10. Проверяем, что если w3 = 0 то символ соответствующий числу не стоит в старшем разряде. Проверяем заняты ли полученные числа и переходим к разряду выше или ниже в зависимости от результата проверки.

В оптимизированном алгоритме существенно уменьшено перебор значений благодаря тому, что в большинстве случаев перебор не требуется и числа рассчитываются исходя из известных, а также подбор следующих значений не начнётся если на предыдущем разряде условия не выполнены.

В таблице 1 представлены код и время выполнения до и после оптимизации. Тестирование проводилось на крипторифме attractions + intentions = regeneration.

*Таблица 1 – Результаты оптимизации алгоритма*

|  |  |  |
| --- | --- | --- |
|  | *До изменений* | *После изменений* |
| *main* | int main()  {  char word1[65] ;  char word2[65] ;  char word3[65] ;  scanf("%s", word1);  scanf("%s", word2);  scanf("%s", word3);  clock\_t begin = clock();  len = strlen(word3);  char\_list\_add\_elem(word1);  char\_list\_add\_elem(word2);  char\_list\_add\_elem(word3);  if (permutation(word1, word2, word3, 0) != 1)  printf("\nрешение не найдено");  printf("\n%f", (double)(clock() - begin) / CLOCKS\_PER\_SEC);  } | int main()  {  char word1[65];  char word2[65];  char word3[65];  scanf("%s", word1);  scanf("%s", word2);  scanf("%s", word3  long begin = clock();  len3 = strlen(word3);  len2 = strlen(word2);  len1 = strlen(word1);  int f;  if (len3 > len2 && len3 > len1) {  ar[1] = word3[0];  if (len2 > len1) {  ar[9] = word2[0];  ar[0] = word3[1];  }  else if (len1 > len2) {  ar[9] = word1[0];  ar[0] = word3[1];  }  }  char buffer1[65];  char buffer2[65];  buffer2[0] = '\0';  buffer1[0] = '\0';  while (len3 != len2 || len2 != len1) {  if (len2 < len1 || len2 < len3) {  strcat(buffer2, " ");  len2++;  }  if (len1 < len2 || len1 < len3) {  strcat(buffer1, " ");  len1++;  }  }  strcat(buffer2, word2);  strcat(buffer1, word1);  f = permutation2(buffer1, buffer2, word3, 0, len3 - 1, 1);  if (f == 1) {  printf("\n%lld", res1);  printf("\n%lld", res2);  printf("\n%lld", res3);  }  else printf("\nрешение не найдено");  printf("\n%f", (double)(clock() - begin) / CLOCKS\_PER\_SEC);  } |
| *struct char\_list*  *и*  *char\_list\_add\_elem* | typedef struct node  {  char c;  int v;  struct node\* next\_elem;  }char\_list;  char\_list arr[10];  int use[10] = { 0,0,0,0,0,0,0,0,0,0 };  int len;  void char\_list\_add\_elem(char\* word)  {  int flag;  for (int i = 0; i < strlen(word); i++) {  flag = 0;  int j = 0;  for (j = 0; j < 10 && arr[j].c != NULL; j++) {  if (arr[j].c == word[i]) {  flag = 1;  break;  }  }  if (flag == 0) {  arr[j].c = word[i];  arr[j].v = -1;  }  }  } | *Структура отсутствует.* |
| *permutation* | int permutation(char\* word1, char\* word2, char\* word3, int k) {  int i;  for (i = 0; i < 10; i++) {  if (use[i] != -1 && (arr[k].c != word1[0] || i != 0) && (arr[k].c != word2[0] || i != 0) && (arr[k].c != word3[0] || i != 0)) {  arr[k].v = i;  use[i] = -1;  if (arr[k + 1].c != NULL && k != 9) {  if (permutation(word1, word2, word3, k + 1) == 1)  return 1;  }  else if (chek(word1, word2, word3) == 1) {  return 1;  }  use[i] = 0;  }  }  arr[k].v = -1;  return 0;  } | unsigned long long res1 = 0, res2 = 0, res3 = 0;  int len1, len2, len3;  char ar[10];  int permutation (char\* word1, char\* word2, char\* word3, int c, int j, unsigned long long k) {  if (j < 0) {  return 1;  }  int w1 = -1, w2 = -1, w3 = -1;  if (word1[j] == ' ')  w1 = 0;  if (word2[j] == ' ')  w2 = 0;  for (int i = 0; i < 10; i++) {  if (word1[j] == ar[i])  w1 = i;  if (word2[j] == ar[i])  w2 = i;  if (word3[j] == ar[i])  w3 = i;  if (w1 != -1 && w2 != -1 && w3 != -1)  break;  }  int f;  if (w3 != -1) {  if (w1 != -1) {  if (w2 != -1) {  if ((w1 + w2 + c) % 10 == w3) {  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w2 \* k;  res3 += w3 \* k;  }  return f;  }  else  return 0;  }  else {  if (w3 - w1 - c >= 0)  w2 = w3 - w1 - c;  else  w2 = w3 - w1 - c + 10;  if (ar[w2] != NULL)  return 0;  ar[w2] = word2[j];  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w2 \* k;  res3 += w3 \* k;  }  else  ar[w2] = NULL;  return f;  }  }  else if (w2 != -1) {  if (w3 - w2 - c >= 0)  w1 = w3 - w2 - c;  else  w1 = w3 - w2 - c + 10;  if (ar[w1] != NULL)  return 0;  ar[w1] = word1[j];  f = permutation2(word1, word2, word3, (w1 + w2) / 10, --j, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w2 \* k;  res3 += w3 \* k;  }  else  ar[w1] = NULL;  return f;  }  else {  if (word1[j] == word2[j]) {  if (c == 0) {  w1 = w3 >> 1;  if (ar[w1] == NULL) {  ar[w1] = word1[j];  f = permutation2(word1, word2, word3, (w1 + w1) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += w3 \* k;  return 1;  }  else  ar[w1] = NULL;    }  w1 = (w3 + 10) >> 1;  if (ar[w1] == NULL) {  ar[w1] = word1[j];  f = permutation2(word1, word2, word3, (w1 + w1) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += w3 \* k;  return 1;  }  else  ar[w1] = NULL;  }  return 0;  }  else {  w1 = --w3 >> 1;  if (ar[w1] == NULL) {  ar[w1] = word1[j];  f = permutation2(word1, word2, word3, (w1 + w1 +1) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += w3 \* k;  return 1;  }  else  ar[w1] = NULL;    }  w1 = (--w3 + 10) >>1;  if (ar[w1] == NULL) {  ar[w1] = word1[j];  f = permutation2(word1, word2, word3, (w1 + w1 + 1) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += w3 \* k;  return 1;  }  else  ar[w1] = NULL;  }  return 0;  }  }  else {  for (int i = 1; i < 10; i++) {  if (ar[i] == NULL) {  w1 = i;  w2 = w3 - i - c;  if (w2 < 0)  w2 = w3 + 10 - i - c;  if (w1 != w2 && ar[w2] == NULL) {  ar[w1] = word1[j];  ar[w2] = word2[j];  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w2 \* k;  res3 += w3 % 10 \* k;  return 1;  }  else {  ar[w1] = NULL;  ar[w2] = NULL;  }  }  }  }  return 0;  }  }  }  else {  if (w1 != -1) {  if (w2 != -1) {  w3 = (w1 + w2 + c) % 10;  if (ar[w3] != NULL)  return 0;  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w2 \* k;  res3 += w3 \* k;  }  else  ar[w3] = NULL;  return f;  }  else {  for (int i = 0; i < 10; i++) {  w3 = (w1 + i + c) % 10;  if (ar[i] == NULL && ar[w3] == NULL) {  ar[i] = word2[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += i \* k;  res3 += w3 \* k;  return 1;  }  else {  ar[i] = NULL;  ar[w3] = NULL;  }  }  }  return 0;  }  }  else if (w2 != -1) {  for (int i = 0; i < 10; i++) {  w3 = (w2 + i + c) % 10;  if (ar[i] == NULL && ar[w3] == NULL) {  ar[i] = word1[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += w2 \* k;  res3 += w3 % 10 \* k;  return 1;  }  else {  ar[i] = NULL;  ar[w3] = NULL;  }  }  }  return 0;  }  else {  if (word1[j] == word2[j] && word1[j] == word3[j]) {  if (c == 0) {  if (ar[0] != NULL)  return 0;  ar[0] = word3[j];  f = permutation2(word1, word2, word3, 0, --j, k \* 10);  if (f == 0)  ar[0] = NULL;  return f;  }  else {  if (ar[9] != NULL)  return 0;  ar[9] = word3[j];  f = permutation2(word1, word2, word3, 1, --j, k \* 10);  if (f == 1) {  res1 += 9 \* k;  res2 += 9 \* k;  res3 += 9 \* k;  }  else  ar[9] = NULL;  return f;  }  }  else if (word1[j] == word3[j]) {  if (c == 0) {  if (ar[0] != NULL)  return 0;  ar[0] = word2[j];  for (int i = 1; i < 10; i++) {  ar[i] = word1[j];  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[0] = NULL;  return 0;  }  else {  if (ar[9] != NULL)  return 0;  ar[9] = word2[j];  for (int i = 1; i < 10; i++) {  ar[i] = word1[j];  f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += 9 \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[9] = NULL;  return 0;  }  }  else if (word2[j] == word3[j]) {  if (c == 0) {  if (ar[0] != NULL)  return 0;  ar[0] = word1[j];  for (int i = 1; i < 10; i++) {  ar[i] = word2[j];  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 1) {  res2 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[0] = NULL;  return 0;  }  else {  if (ar[9] != NULL)  return 0;  ar[9] = word1[j];  for (int i = 1; i < 10; i++) {  ar[i] = word2[j];  f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);  if (f == 1) {  res1 += 9 \* k;  res2 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[9] = NULL;  return 0;  }  }  else if (word2[j] == word1[j]) {  if (c == 0) {  for (int i = 2; i < 20; i += 2) {  w1 = i >> 1;  w3 = i % 10;  if (ar[w3] == NULL && ar[w1] == NULL)  {  ar[w1] = word1[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, i / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += i % 10 \* k;  return 1;  }  else {  ar[w1] = NULL;  ar[w3] = NULL;  }  }  }  return 0;  }  else {  for (int i = 1; i < 20; i += 2) {  w1 = (i-1) >> 1;  w3 = i % 10;  if (ar[w3] == NULL && ar[w1] == NULL) {  ar[w1] = word1[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (i - 1) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += w3 \* k;  return 1;  }  else {  ar[w1] = NULL;  ar[w3] = NULL;  }  }  }  return 0;  }  }  else {  for (int i = 1; i < 10; i++) {  if (ar[i] == NULL) {  ar[i] = word1[j];  for (int r = 1; r < 10 && i != r; r++) {  if (ar[r] == NULL && ar[(r + i + c) % 10] == NULL ) {  if ((r + i + c) % 10 != 0 || (word3[j] != word1[0] && word3[j] != word2[0])) {  ar[r] = word2[j];  ar[(r + i + c) % 10] = word3[j];  f = permutation2(word1, word2, word3, (r + i + c) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += r \* k;  res3 += (r + i + c) % 10 \* k;  return 1;  }  else {  ar[r] = NULL;  ar[(r + i + c) % 10] = NULL;  }    }  }  }  ar[i] = NULL;  }  }  return 0;  }  }  }  } |
| *chek* | int chek(char\* word1, char\* word2, char\* word3) {  long long w1 = 0, w2 = 0, w3 = 0;  int f;  for (int i = 0; i < len; i++) {  f = 0;  for (int j = 0; j < 10 && arr[j].c != NULL; j++) {  if (arr[j].c == word1[i]) {  w1 = w1 \* 10 + (arr[j].v);  f++;  }  if (arr[j].c == word2[i]) {  w2 = w2 \* 10 + arr[j].v;  f++;  }  if (arr[j].c == word3[i]) {  w3 = w3 \* 10 + arr[j].v;  f++;  }  if (f == 3)  break;  }  }  if (w1 + w2 == w3) {  printf("\n%lld", w1);  printf("\n%lld", w2);  printf("\n%lld", w3);  return 1;  }  else return 0;  } | *Функция отсутствует так, как проверка осуществляется во время подбора.* |
| *Время* | 1,019 сек | < 0,001 сек |
| 1894823822 тиков | 1681425 тиков |

**Коэффициент ускорения:** 1126

Так как функция clock() возвращает значение времени в миллисекундах, а время выполнения программы после оптимизации меньше 1 миллисекунды. Измерить время в секундах невозможно, дальше измерения будут проводиться в тиках.

МАШИННО-НЕЗАВИСИМАЯ ОПТИМИЗАЦИЯ

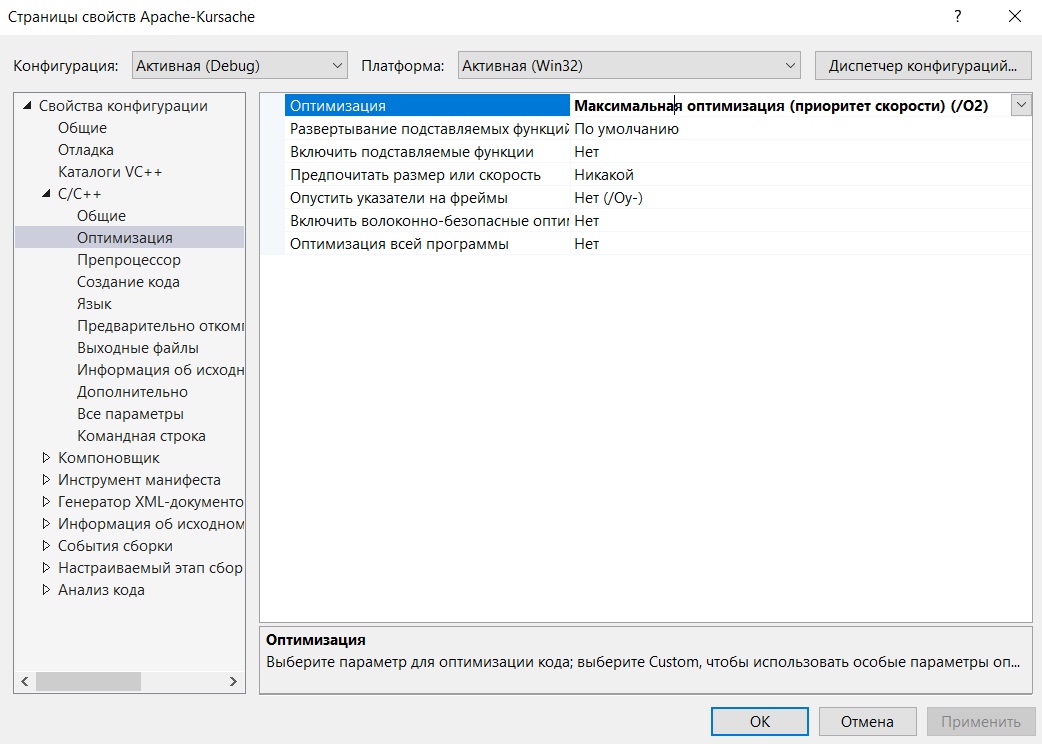
*Таблица 2 – Результаты машинно-независимой оптимизации*

|  |  |  |
| --- | --- | --- |
|  | *До изменений* | *После изменений* |
| *Снижение мощности* | ...  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, j-1, k \* 10);  ...  w1 = w3 /2;  ...  w1 = (w3 + 10) /2;  ...  w1 = (w3-1) /2;  ...  w1 = (w3 + 9) /2;  ...  w1 = i / 2;  ...  w1 = (i-1) / 2;  ...  f = permutation2(word1, word2, word3, (w1\*2 + c) / 10, j - 1, k \* 10);  … | …  Если на в этом вызове функции j больше не используется замена j-1 на –j при вызове функции  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);  ...  Замена /2 на >>1.  w1 = w3 >> 1;  ...  w1 = (w3 + 10) >> 1;  ...  w1 = (w3-1) >> 1;  ...  w1 = (w3 + 9) >>1;  ...  w1 = i >> 1;  ...  w1 = (i-1) >> 1;  ...  Замена \*2 на <<1, так как известно, что с=0, +с не выполняем.  f = permutation2(word1, word2, word3, (w1<<1) / 10, j - 1, k \* 10);  ...  Замена \*2 на <<1, так как известно, что с=1, заменяем +с на +1.  f = permutation2(word1, word2, word3, (w1<<1 +1) / 10, j - 1, k \* 10);  ... |
| *Избавление от лишних операций* | …  res1+=w1\*k;  res2+=w2\*k;  res3+=w3\*k;  f=permutation2(word1, word2, word3, (w1+w2+c)/10, --j, k\*10);  if(f==0){ res1-=w1\*k;  res2-=w2\*k;  res3-=w3\*k;  }  return f;  …  w1=i;  w2=w3-i-c;  if (w2<0)  w2=w3+10-i-c;  if(w1!=w2&&ar[w2]==NULL){  ar[w1]=word1[j];  ar[w2]=word2[j];  res1+=w1\*k; res2+=w2\*k; res3+=w3%10\*k; f=permutation2(word1, word2, word3, (w1+w2+c)/10, j-1, k\*10);  if(f==0){  ar[w1]=NULL;  ar[w2]=NULL;  res1-=w1\*k;  res2-=w2\*k;  res3-=w3%10\*k;  }  else  return f;  …  if(ar[i]==NULL&&ar[(w1+i+c)%10]==NULL){  ar[i]=word2[j];  ar[(w1+i+c)%10]=word3[j];  res1+=w1\*k;  res2+=i\*k; res3+=(w1+i+c)%10\*k;  f=permutation2(word1, word2, word3, (w1+i+c)/10, j-1, k\*10);  if(f==0){  ar[i]=NULL;  ar[(w1+i+c)%10]=NULL;  res1-=w1\*k;  res2-=i\*k;  res3-=(w1+i+c)%10\*k;  }else  return f;  }  …  f=permutation2(word1, word2, word3, (w1+w2+c)/10, --j, k\*10);  …  ar[0]=word2[j];  ar[i]=word1[j];  res1+=i\*k;  res3+=i\*k; f=permutation2(word1, word2, word3, 0, j-1, k\*10);  if(f==0){  ar[0]=NULL;  ar[i]=NULL;  res1-=i\*k;  res3-=i\*k;  }  …  ar[i/2]=word1[j];  ar[i%10]=word3[j];  res1+=i/2\*k; res2+=i/2\*k; res3+=i%10\*k; f=permutation2(word1, word2, word3, i/10, j-1, k\*10);  if(f==0){  ar[i/2]=NULL;  ar[i%10]=NULL; res1-=i/2\*k; res2-=i/2\*k; res3-=i%10\*k;  }  else  return f;  … | …  f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w2 \* k;  res3 += w3 \* k;  }  return f;  …  w2 = w3 - i - c;  if (w2 < 0)  w2 = w3 + 10 - i - c;  if (i != w2 && ar[w2] == NULL) {  ar[i] = word1[j];  ar[w2] = word2[j];  f = permutation2(word1, word2, word3, (i + w2 + c) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += w2 \* k;  res3 += w3 % 10 \* k;  return 1;  }  else {  ar[i] = NULL;  ar[w2] = NULL;  }  }  …  w3 = (w1 + i + c) % 10;  if (ar[i] == NULL && ar[w3] == NULL) {  ar[i] = word2[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += i \* k;  res3 += w3 \* k;  return 1;  }  else {  ar[i] = NULL;  ar[w3] = NULL;  }  }  …  Не выполняем расчёт (w1+w2+c)/10 там, где известно, чему равен результат.  f=permutation2(word1, word2, word3, 0, --j, k\*10);  …  f=permutation2(word1, word2, word3, 1, --j, k\*10);  …  ar[i] = word1[j];  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  …  w1 = i >> 1;  w3 = i % 10;  if (ar[w3] == NULL && ar[w1] == NULL) {  ar[w1] = word1[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, i / 10, j - 1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += w1 \* k;  res3 += i % 10 \* k;  return 1;  }  else {  ar[w1] = NULL;  ar[w3] = NULL;  }  }  … |
| *Вынос инвариантного кода* | …  for (int i = 0; i < 10; i++) {  if (ar[i] == NULL && ar[(w1 + i + c) % 10] == NULL) {  ar[i] = word2[j];  ar[(w1 + i + c) % 10] = word3[j];  res1 += w1 \* k;  res2 += i \* k;  res3 += (w1 + i + c) % 10 \* k;  f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);  if (f == 0) {  ar[i] = NULL;  ar[(w1 + i + c) % 10] = NULL;  res1 -= w1 \* k;  res2 -= i \* k;  res3 -= (w1 + i + c) % 10 \* k;  }  else return f;  }  }  …  for (int i = 0; i < 10; i++) {  if (ar[i] == NULL && ar[(w2 + i + c) % 10] == NULL) {  ar[i] = word1[j];  ar[(w2 + i + c) % 10] = word3[j];  res1 += i \* k;  res2 += w2 \* k;  res3 += (w2 + i + c) % 10 \* k;  f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);  if (f == 0) {  ar[i] = NULL;  ar[(w2 + i + c) % 10] = NULL;  res1 -= i \* k;  res2 -= w2 \* k;  res3 -= (w2 + i + c) % 10 \* k;  }  else return f;  }  }  …  for (int i = 1; i < 10; i++) {  ar[0] = word2[j];  ar[i] = word1[j];  res1 += i \* k;  res3 += i \* k;  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 0) {  [0] = NULL;  ar[i] = NULL;  res1 -= i \* k;  res3 -= i \* k;  }  else  return f;  }  …  for (int i = 1; i < 10; i++) {  ar[9] = word2[j];  ar[i] = word1[j];  res1 += i \* k;  res2 += 9 \* k;  res3 += i \* k;  f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);  if (f == 0) {  ar[9] = NULL;  ar[i] = NULL;  res1 -= i \* k;  res2 -= 9 \* k;  res3 -= i \* k;  }  else  return f;  }  …  for (int i = 1; i < 10; i++) {  ar[0] = word1[j];  ar[i] = word2[j];  res2 += i \* k;  res3 += i \* k;  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 0) {  ar[0] = NULL;  ar[i] = NULL;  res2 -= i \* k;  res3 -= i \* k;  }  else  return f;  }  …  for (int i = 1; i < 10; i++) {  ar[9] = word1[j];  ar[i] = word2[j];  res1 += i \* k;  res2 += 9 \* k;  res3 += i \* k;  f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);  if (f == 0) {  ar[9] = NULL;  ar[i] = NULL;  res1 -= i \* k;  res2 -= 9 \* k;  res3 -= i \* k;  }  else  return f;  }  …  for (int r = 1; r < 10; r++) {  if (ar[r] == NULL && ar[(r + i + c) % 10] == NULL && i != r) {  if ((r + i + c) % 10 != 0 || (word3[j] != word1[0] && word3[j] != word2[0])) {  ar[i] = word1[j];  ar[r] = word2[j];  ar[(r + i + c) % 10] = word3[j];  res2 += r \* k;  res3 += (r + i + c) % 10 \* k;  res1 += i \* k;  f = permutation2(word1, word2, word3, (r + i + c) / 10, j - 1, k \* 10);  if (f == 0) {  ar[i] = NULL;  ar[r] = NULL;  ar[(r + i + c) % 10] = NULL;  res1 -= i \* k;  res2 -= r \* k;  res3 -= (r + i) % 10 \* k;  }  else  return f;  }  }  }  … | …  int sum = w1 + c;//Вынос инварианта  for (int i = 0; i < 10; i++) {  w3 = (i + sum) % 10;  if (ar[i] == NULL && ar[w3] == NULL) {  ar[i] = word2[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (i + sum) / 10, j-1, k \* 10);  if (f == 1) {  res1 += w1 \* k;  res2 += i \* k;  res3 += w3 \* k;  return 1;  }  else {  ar[i] = NULL;  ar[w3] = NULL;  }  }  }  …  int sum = w2 + c;//Вынос инварианта  for (int i = 0; i < 10; i++) {  w3 = (i + sum) % 10;  if (ar[i] == NULL && ar[w3] == NULL) {  ar[i] = word1[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (i + sum) / 10, j-1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += w2 \* k;  res3 += w3 % 10 \* k;  return 1;  }  else {  ar[i] = NULL;  ar[w3] = NULL;  }  }  }  …  ar[0] = word2[j];//Вынос инварианта  for (int i = 1; i < 10; i++) {  ar[i] = word1[j];  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[0] = NULL;//Вынос инварианта  …  ar[9] = word2[j];//Вынос инварианта  for (int i = 1; i < 10; i++) {  ar[i] = word1[j];  f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += 9 \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[9] = NULL;//Вынос инварианта  …  ar[0] = word1[j];//Вынос инварианта  for (int i = 1; i < 10; i++) {  ar[i] = word2[j];  f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);  if (f == 1) {  res2 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[0] = NULL;//Вынос инварианта  …  ar[9] = word1[j];//Вынос инварианта  for (int i = 1; i < 10; i++) {  ar[i] = word2[j];  f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);  if (f == 1) {  res1 += 9 \* k;  res2 += i \* k;  res3 += i \* k;  return 1;  }  else  ar[i] = NULL;  }  ar[9] = NULL;//Вынос инварианта  …  ar[i] = word1[j];//Вынос инварианта  sum = i + c;//Вынос инварианта  for (int r = 1; r < 10 && i != r; r++) {  w3 = (r + sum) % 10;  if (ar[r] == NULL && ar[w3] == NULL ) {  if (w3 != 0 || (word3[j] != word1[0] && word3[j] != word2[0])) {  ar[r] = word2[j];  ar[w3] = word3[j];  f = permutation2(word1, word2, word3, (r + sum) / 10, j - 1, k \* 10);  if (f == 1) {  res1 += i \* k;  res2 += r \* k;  res3 += w3 \* k;  return 1;  }  else {  ar[r] = NULL;  ar[w3] = NULL;  }  }  }  }  ar[i] = NULL;//Вынос инварианта  … |
| *Время* | 1681425 тиков | 502681 тиков |

**Коэффициент ускорения:** 3,3

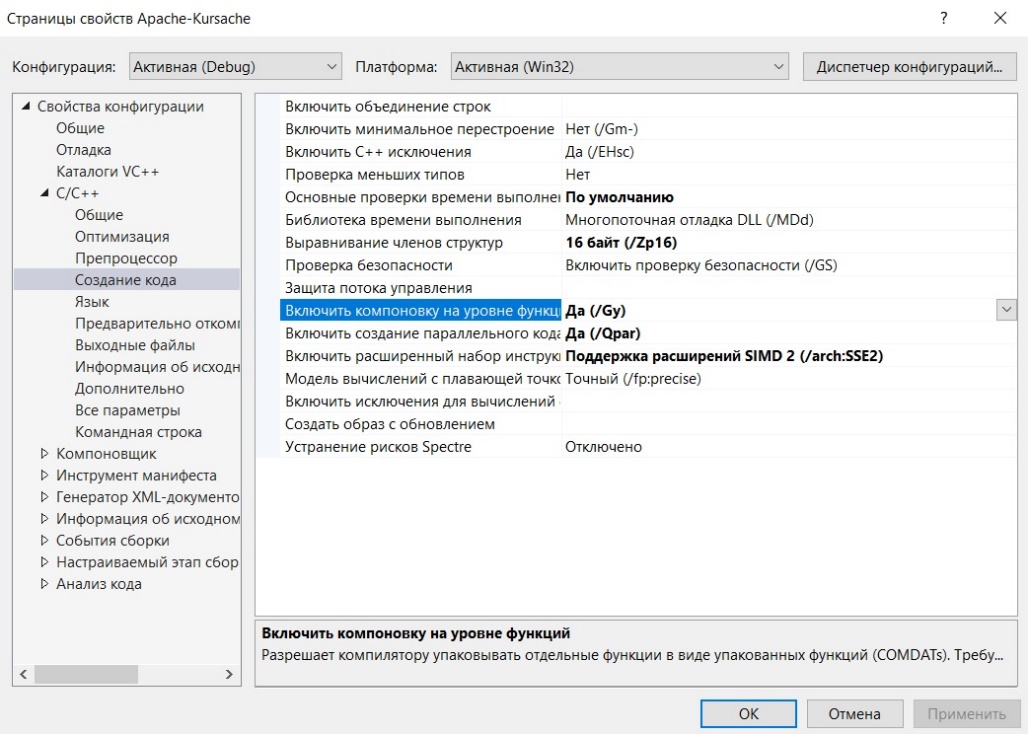
МАШИННО-ЗАВИСИМЫЕ ОПТИМИЗАЦИИ

1. В среде разработки Visual Studio 2019 в свойствах проекта была включена оптимизация кода по скорости /О2 (Рисунок 2).



*Рисунок 2 –раздел Оптимизация*

1. В разделе Создание кода были включены такие настройки, как Выравнивание членов структур 16 байт (/Zp16), Поддержка расширений SIMD 2 (/arch: SSE2) и Создание параллельного кода (/Qpar) (Рисунок 3)



*Рисунок 3 –раздел Создание кода*

*Таблица 3 – Результаты после машинно-зависимой оптимизации*

|  |  |  |
| --- | --- | --- |
|  | *До изменений* | *После изменений* |
| *Время, сек* | 502681 тиков | 287930 тиков |

**Коэффициент ускорения:** 1,7

ВЫВОДЫ

*Таблица 4 – Итог*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *До* | *После* | *Коэффициент* |
| *Время* | 1,019 сек | < 0,001 сек | 6580 |
| 1894823822 тиков | 287930 тиков |
| *Объем* | 64,5 КБ | 55,5 КБ | 1,16 |

Проведя анализ оптимизаций на разных уровнях, можно сделать вывод, что наименее существенный вклад вносит оптимизация программы на машинно-зависимом уровне.

В результате проведённых исследований было установлено, что оптимизация играет немаловажную роль в жизненном цикле проекта, так как после оптимизации реализованная программа работает в несколько раз быстрее, что, безусловно, делает проект более востребованным пользователями.

ПРИЛОЖЕНИЕ 1

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <intrin.h>

typedef struct node

{

char c;

int v;

struct node\* next\_elem;

}char\_list;

char\_list arr[10];

int use[10] = { 0,0,0,0,0,0,0,0,0,0 };

int len;

void char\_list\_add\_elem(char\* word)

{

int flag;

for (int i = 0; i < strlen(word); i++) {

flag = 0;

int j = 0;

for (j = 0; j < 10 && arr[j].c != NULL; j++) {

if (arr[j].c == word[i]) {

flag = 1;

break;

}

}

if (flag == 0) {

arr[j].c = word[i];

arr[j].v = -1;

}

}

}

int chek(char\* word1, char\* word2, char\* word3) {

long long w1 = 0, w2 = 0, w3 = 0;

int f;

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

f = 0;

for (int j = 0; j < 10 && arr[j].c != NULL; j++) {

if (arr[j].c == word1[i]) {

w1 = w1 \* 10 + (arr[j].v);

f++;

}

if (arr[j].c == word2[i]) {

w2 = w2 \* 10 + arr[j].v;

f++;

}

if (arr[j].c == word3[i]) {

w3 = w3 \* 10 + arr[j].v;

f++;

}

if (f == 3)

break;

}

}

if (w1 + w2 == w3) {

printf("\n%lld", w1);

printf("\n%lld", w2);

printf("\n%lld", w3);

return 1;

}

else return 0;

}

int permutation(char\* word1, char\* word2, char\* word3, int k) {

int i;

for (i = 0; i < 10; i++) {

if (use[i] != -1 && (arr[k].c != word1[0] || i != 0) && (arr[k].c != word2[0] || i != 0) && (arr[k].c != word3[0] || i != 0)) {

arr[k].v = i;

use[i] = -1;

if (arr[k + 1].c != NULL && k != 9) {

if (permutation(word1, word2, word3, k + 1) == 1)

return 1;

}

else if (chek(word1, word2, word3) == 1) {

return 1;

}

use[i] = 0;

}

}

arr[k].v = -1;

return 0;

}

int main()

{

char word1[65];

char word2[65];

char word3[65];

scanf("%s", word1);

scanf("%s", word2);

scanf("%s", word3);

clock\_t begin = clock();

len = strlen(word3);

char\_list\_add\_elem(word1);

char\_list\_add\_elem(word2);

char\_list\_add\_elem(word3);

if (permutation(word1, word2, word3, 0) != 1)

printf("\nрешение не найдено");

printf("\n%f\n", (double)(clock() - begin) / CLOCKS\_PER\_SEC);

}

**ПРИЛОЖЕНИЕ 2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <time.h>

#include <intrin.h>

unsigned long long res1 = 0, res2 = 0, res3 = 0;

int len1, len2, len3;

char ar[10];

int permutation2(char\* word1, char\* word2, char\* word3, int c, int j, unsigned long long k) {

if (j < 0) {

return 1;

}

int w1 = -1, w2 = -1, w3 = -1;

if (word1[j] == ' ')

w1 = 0;

if (word2[j] == ' ')

w2 = 0;

for (int i = 0; i < 10; i++) {

if (word1[j] == ar[i])

w1 = i;

if (word2[j] == ar[i])

w2 = i;

if (word3[j] == ar[i])

w3 = i;

if (w1 != -1 && w2 != -1 && w3 != -1)

break;

}

int f;

if (w3 != -1) {

if (w1 != -1) {

if (w2 != -1) {

if ((w1 + w2 + c) % 10 == w3) {

f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w2 \* k;

res3 += w3 \* k;

}

return f;

}

else

return 0;

}

else {

if (w3 - w1 - c >= 0)

w2 = w3 - w1 - c;

else

w2 = w3 - w1 - c + 10;

if (ar[w2] != NULL)

return 0;

ar[w2] = word2[j];

f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w2 \* k;

res3 += w3 \* k;

}

else

ar[w2] = NULL;

return f;

}

}

else if (w2 != -1) {

if (w3 - w2 - c >= 0)

w1 = w3 - w2 - c;

else

w1 = w3 - w2 - c + 10;

if (ar[w1] != NULL)

return 0;

ar[w1] = word1[j];

f = permutation2(word1, word2, word3, (w1 + w2) / 10, --j, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w2 \* k;

res3 += w3 \* k;

}

else

ar[w1] = NULL;

return f;

}

else {

if (word1[j] == word2[j]) {

if (c == 0) {

w1 = w3 >> 1;

if (ar[w1] == NULL) {

ar[w1] = word1[j];

f = permutation2(word1, word2, word3, (w1<<1) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w1 \* k;

res3 += w3 \* k;

return 1;

}

else

ar[w1] = NULL;

}

w1 = (w3 + 10) >> 1;

if (ar[w1] == NULL) {

ar[w1] = word1[j];

f = permutation2(word1, word2, word3, (w1 <<1) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w1 \* k;

res3 += w3 \* k;

return 1;

}

else

ar[w1] = NULL;

}

return 0;

}

else {

w1 = (w3-1) >> 1;

if (ar[w1] == NULL) {

ar[w1] = word1[j];

f = permutation2(word1, word2, word3, (w1<<1 +1) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w1 \* k;

res3 += w3 \* k;

return 1;

}

else

ar[w1] = NULL;

}

w1 = (w3 + 9) >>1;

if (ar[w1] == NULL) {

ar[w1] = word1[j];

f = permutation2(word1, word2, word3, (w1<<1 + 1) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w1 \* k;

res3 += w3 \* k;

return 1;

}

else

ar[w1] = NULL;

}

return 0;

}

}

else {

for (int i = 1; i < 10; i++) {

if (ar[i] == NULL) {

w2 = w3 - i - c;

if (w2 < 0)

w2 = w3 + 10 - i - c;

if (i != w2 && ar[w2] == NULL) {

ar[i] = word1[j];

ar[w2] = word2[j];

f = permutation2(word1, word2, word3, (i + w2 + c) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += i \* k;

res2 += w2 \* k;

res3 += w3 % 10 \* k;

return 1;

}

else {

ar[i] = NULL;

ar[w2] = NULL;

}

}

}

}

return 0;

}

}

}

else {

if (w1 != -1) {

if (w2 != -1) {

w3 = (w1 + w2 + c) % 10;

if (ar[w3] != NULL)

return 0;

ar[w3] = word3[j];

f = permutation2(word1, word2, word3, (w1 + w2 + c) / 10, --j, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w2 \* k;

res3 += w3 \* k;

}

else

ar[w3] = NULL;

return f;

}

else {

for (int i = 0; i < 10; i++) {

w3 = (w1 + i + c) % 10;

if (ar[i] == NULL && ar[w3] == NULL) {

ar[i] = word2[j];

ar[w3] = word3[j];

f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += i \* k;

res3 += w3 \* k;

return 1;

}

else {

ar[i] = NULL;

ar[w3] = NULL;

}

}

}

return 0;

}

}

else if (w2 != -1) {

for (int i = 0; i < 10; i++) {

w3 = (w2 + i + c) % 10;

if (ar[i] == NULL && ar[w3] == NULL) {

ar[i] = word1[j];

ar[w3] = word3[j];

f = permutation2(word1, word2, word3, (w1 + i + c) / 10, j-1, k \* 10);

if (f == 1) {

res1 += i \* k;

res2 += w2 \* k;

res3 += w3 % 10 \* k;

return 1;

}

else {

ar[i] = NULL;

ar[w3] = NULL;

}

}

}

return 0;

}

else {

if (word1[j] == word2[j] && word1[j] == word3[j]) {

if (c == 0) {

if (ar[0] != NULL)

return 0;

ar[0] = word3[j];

f = permutation2(word1, word2, word3, 0, --j, k \* 10);

if (f == 0)

ar[0] = NULL;

return f;

}

else {

if (ar[9] != NULL)

return 0;

ar[9] = word3[j];

f = permutation2(word1, word2, word3, 1, --j, k \* 10);

if (f == 1) {

res1 += 9 \* k;

res2 += 9 \* k;

res3 += 9 \* k;

}

else

ar[9] = NULL;

return f;

}

}

else if (word1[j] == word3[j]) {

if (c == 0) {

if (ar[0] != NULL)

return 0;

ar[0] = word2[j];

for (int i = 1; i < 10; i++) {

ar[i] = word1[j];

f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);

if (f == 1) {

res1 += i \* k;

res3 += i \* k;

return 1;

}

else

ar[i] = NULL;

}

ar[0] = NULL;

return 0;

}

else {

if (ar[9] != NULL)

return 0;

ar[9] = word2[j];

for (int i = 1; i < 10; i++) {

ar[i] = word1[j];

f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);

if (f == 1) {

res1 += i \* k;

res2 += 9 \* k;

res3 += i \* k;

return 1;

}

else

ar[i] = NULL;

}

ar[9] = NULL;

return 0;

}

}

else if (word2[j] == word3[j]) {

if (c == 0) {

if (ar[0] != NULL)

return 0;

ar[0] = word1[j];

for (int i = 1; i < 10; i++) {

ar[i] = word2[j];

f = permutation2(word1, word2, word3, 0, j - 1, k \* 10);

if (f == 1) {

res2 += i \* k;

res3 += i \* k;

return 1;

}

else

ar[i] = NULL;

}

ar[0] = NULL;

return 0;

}

else {

if (ar[9] != NULL)

return 0;

ar[9] = word1[j];

for (int i = 1; i < 10; i++) {

ar[i] = word2[j];

f = permutation2(word1, word2, word3, 1, j - 1, k \* 10);

if (f == 1) {

res1 += 9 \* k;

res2 += i \* k;

res3 += i \* k;

return 1;

}

else

ar[i] = NULL;

}

ar[9] = NULL;

return 0;

}

}

else if (word2[j] == word1[j]) {

if (c == 0) {

for (int i = 2; i < 20; i += 2) {

w1 = i >> 1;

w3 = i % 10;

if (ar[w3] == NULL && ar[w1] == NULL) {

ar[w1] = word1[j];

ar[w3] = word3[j];

f = permutation2(word1, word2, word3, i / 10, j - 1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w1 \* k;

res3 += i % 10 \* k;

return 1;

}

else {

ar[w1] = NULL;

ar[w3] = NULL;

}

}

}

return 0;

}

else {

for (int i = 1; i < 20; i += 2) {

w1 = (i-1) >> 1;

w3 = i % 10;

if (ar[w3] == NULL && ar[w1] == NULL) {

ar[w1] = word1[j];

ar[w3] = word3[j];

f = permutation2(word1, word2, word3, (i - 1) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += w1 \* k;

res2 += w1 \* k;

res3 += w3 \* k;

return 1;

}

else {

ar[w1] = NULL;

ar[w3] = NULL;

}

}

}

return 0;

}

}

else {

for (int i = 1; i < 10; i++) {

if (ar[i] == NULL) {

ar[i] = word1[j];

for (int r = 1; r < 10 && i != r; r++) {

if (ar[r] == NULL && ar[(r + i + c) % 10] == NULL ) {

if ((r + i + c) % 10 != 0 || (word3[j] != word1[0] && word3[j] != word2[0])) {

ar[r] = word2[j];

ar[(r + i + c) % 10] = word3[j];

f = permutation2(word1, word2, word3, (r + i + c) / 10, j - 1, k \* 10);

if (f == 1) {

res1 += i \* k;

res2 += r \* k;

res3 += (r + i + c) % 10 \* k;

return 1;

}

else {

ar[r] = NULL;

ar[(r + i + c) % 10] = NULL;

}

}

}

}

ar[i] = NULL;

}

}

return 0;

}

}

}

}

int main()

{

char word1[65];

char word2[65];

char word3[65];

scanf("%s", word1);

scanf("%s", word2);

scanf("%s", word3);

long long begin = \_\_rdtsc();

len3 = strlen(word3);

len2 = strlen(word2);

len1 = strlen(word1);

int f;

if (len3 > len2 && len3 > len1) {

ar[1] = word3[0];

if (len2 > len1) {

ar[9] = word2[0];

ar[0] = word3[1];

}

else if (len1 > len2) {

ar[9] = word1[0];

ar[0] = word3[1];

}

}

char buffer1[65];

char buffer2[65];

buffer2[0] = '\0';

buffer1[0] = '\0';

while (len3 != len2 || len2 != len1) {

if (len2 < len1 || len2 < len3) {

strcat(buffer2, " ");

len2++;

}

if (len1 < len2 || len1 < len3) {

strcat(buffer1, " ");

len1++;

}

}

strcat(buffer2, word2);

strcat(buffer1, word1);

f = permutation2(buffer1, buffer2, word3, 0, len3 - 1, 1);

if (f == 1) {

printf("\n%lld", res1);

printf("\n%lld", res2);

printf("\n%lld", res3);

}

else printf("\nрешение не найдено");

printf\_s("%I64d ticks\n", \_\_rdtsc() - begin);

}