**Оглавление:**

Оглавление…...………………………………………………………………………..3

Введение…………………………………………………………………………….…4

Основная часть……..………………………………………………………………….6

Код программы……………………………………………………….………..6

Фрагменты работы программы……………………………………………....11

Заключение……………………………………………………………………………12

Список используемой литературы…………………………………………...13

Приложение…………………………………………………………………...14

**Введение**

Современный мир. Цифровые технологии давно заняли одну из самых главных позиций не только в программировании в частности, но и во всей человеческой жизни и постепенно вытесняют собой все остальные виды передачи и хранения информации. Человек понимает – это удобно, компактно, быстро и предельно универсально. Кто-то против подобного развития, кто-то сам всеми силами продвигает его в сторону нашего будущего, но, к сожалению, или к счастью, от нас это уже практически не зависит.

Одним из подобных аспектов технологического развития являются языки программирования и C в частности. За отраслью «кодинга» стоит чуть ли не половина современной жизни, включая дизайн, игровую индустрию, кино, мультипликацию, рекламу и мультимедиа, развлекательные и обучающие программы, архитектуру, промышленное производство и так далее. Это показывает, что важные сферы общества твердо держатся на основе цифровых достижений человечества, и давно не представляются без этого прочного основания.

Целью в данной исследовательской работе будет разработка калькулятора.

Для достижения данной цели понадобится ряд поставленных задач, выполнение которых поможет плавно перейти к сути исследования:

* Изучить синтаксис языка С.
* Рассмотреть простейшие логические конструкции и проиллюстрировать способы их применения.
* Изучить примеры профессиональных работ, использующихся в создании реальных проектов.
* Ознакомиться с подходящей средой программирования.
* Подвести итоги по средствам цельных выводов и мнения о проведенном исследовании.

Задача курсовой работы состоит в разработке программы - калькулятора, которая позволяет считать выражения, состоящие из чисел, тригонометрических выражений и логарифмов. Необходимо разработать удобный пользовательский интерфейс для работы с программой.

Для реализации данной задачи удобно использовать перевод математического выражения из привычной записи в обратную польскую запись. Данный алгоритм подробно описан в курсовой работе напарника.

Для удобной реализации перевода математического выражения из привычной записи в обратную польскую запись удобно использовать динамическую структуру данных. В работе были использованы динамически растущие стеки на односвязном списке.

Стек — абстрактный тип данных, представляющий собой список элементов, организованных по принципу *LIFO* ( «последним пришёл — первым вышел»).

Односвязный список состоит из узлов. Каждый узел содержит значение и указатель на следующий узел. Вначале списка нет и указатель ссылается на NULL.   
Для добавления нового узла необходимо

* 1) Выделить под него память.
* 2) Задать ему значение
* 3) Сделать так, чтобы он ссылался на предыдущий элемент (или на NULL, если его не было)
* 4) Перекинуть указатель вершины на новый узел.

Так описывается стандартная функция стека “push”. Данная функция нужна для внесения на вершину стека, основанного на списке, значение Теперь рассмотрим функцию «pop»: она удаляет элемент, на который указывает вершина и возвращает его значение. Эта функция нужна для возвращения вершины стека и назначения следующего элемента вершиной стека.

Так же стек можно было реализовать на массиве фиксированного значения, что не подходит для нашей задачи, ведь это сделает работу с памятью не рациональной, и у пользователя будут трудности в работе с программой.

Стек, реализованный на динамически растущем массиве тоже не очень подходит для решения поставленной задачи, потому что максимальный размер стека может быть ограничен каким-то числом, либо размером оперативной памяти. Поэтому самым рациональным вариантом является стек, реализованный на односвязном списке.

Программу разработана на языке C в среде Microsoft Visual Studio. Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных.

**Основная часть**

**Код программы**

Программа была выполнена двумя студентами. Разбираемая часть программы показывает работы динамических структур, в частности стэков, которые основаны на списках. Так же данная часть программы показывает функцию логирования в программе и реализацию интерфейса для оптимизации работы с памятью и для удобства работы пользователя. Так же показано, как программа заносит в разные динамические структуры как численные символы, так и буквенные и остальные.

В программе использовались следующие библиотеки:

1. **stdio.h -** заголовочный файл стандартной библиотеки языка Си, содержащий определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода.
2. **math.h -** заголовочный файл стандартной библиотеки языка програмирования Си, разработанный для выполнения простых математических операций.
3. **string.h -** заголовочный файл стандартной библиотеки языка програмирования Си, содержащий функции для работы с нуль-терминированными строками и различными функциями работы с памятью.
4. **locale.h -** заголовочный файл стандартной библиотеки языка програмирования Си, который используется для задач, связанных с локализацией.

|  |
| --- |
|  |

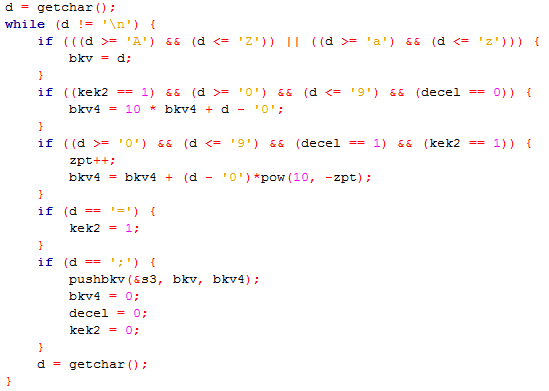
Блок программы 1. Объявление структур

В данном блоке программы идёт объявление динамической структуры данных – стэка. В программе используются 3 стэка для хранения данных. Первый для хранения чисел, второй для хранения знаков, третия для хранения букв и их значений. Стэки состоят из динамичских списков и содержат переменные и указатели на следующую переменную.

|  |
| --- |
|  |

Блок программы 2. Функции для работы со стеками

Здесь идет описание функции, совершающие разные действия со стэками. Функция «peak» возвращает значение вершины стэка чисел, «peakchar» возвращает приоритет от текущего знака, используя функцию «prior». Функция «getsize» возвращает количество элементов стэка (размер стэка). Функция «found» возвращает значение буквы. Функции, начинающиеся на «pop» возвращают вершину стэка, убирая это значение из стэка, и назначают вешиной следующий элемент стэка с помощь вспомогательного стэка. Функции, начинающиеся на «push» заносят на вершину стэка данное значение и создают указатель на предыдущий элемент стэка при помощи вспомогательного стэка.



Блок программы 3. Работа с переменными калькулятора и их значениями

В данном блоке программы идет работа с переменными калькулятора и их значениями. Если введённый знак “;”, тогда программа пишет в файл введённые буквы и кладёт в стэк букву и её значение. Далее обнуляются переменные, для работы со следующей буквой и её знчением.

|  |
| --- |
|  |

Блок программы 4. Дальнейшая работа с буквами и работа с тригонометрическими функциями и логарифмами.

В данном блоке программы идет работа с буквами и тригонометрическими функциями и логарифмами. Если в ведённой строке данный символ является буквой любого регистра, то в нулевую ячейку массива заносится значение введённой буквы. Далее идёт проверка на тригонометрическую функцию. Если второй введённый сивол тоже буква, тогда в массив заносится эта тригонометрическая функция или логарифм. Следующая ячейка равна сиволу нуля. Далее программа в зависимости от содержания массива записывает в файл тригонометрическую функцию или логарифм и присваивает переменной символ, обозначающий данную функцию или логарифм. Если введёныый символ простая переменная, тогда программа выводит, что данные введены корректно и присваивает переменной чисел значение буквы (значение введённой переменной).

|  |
| --- |
|  |

Блок программы 5. Вычисления..

В данном блоке программы посимвольно собираются числовые значения. В зависимости от значения переменной «decel» , переменной «kek1» присваиваются дробное, либо целое значение. В этом же блоке определяется правильный порядок действий, производятся расчеты каждого действия, благодаря функции «math», которая в зависимости «stackChar» определяет знак, производит вычисления, и выодит временное значение веденного выражения.

**Фрагменты работы программы**

Рисунок 1.

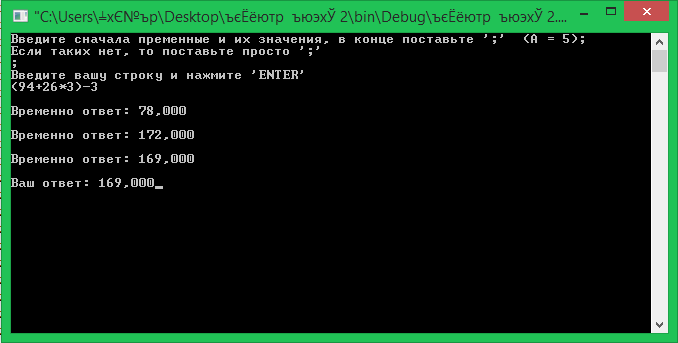
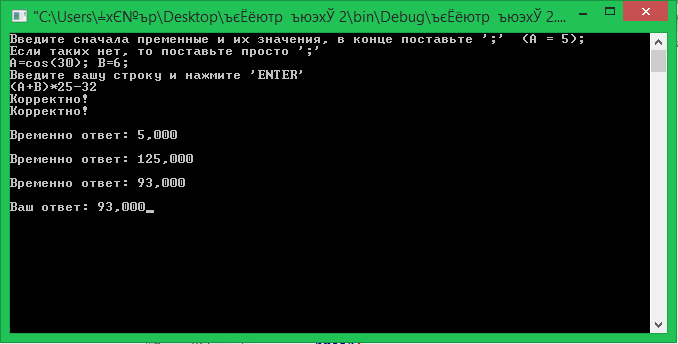


Рисунок 2.



Рисунок 3.



**Заключение**

Цель работы, а именно – создание программы для решения арифметических выражений, была успешно выполнена. Код программы продемонстрировал свою работоспособность в решении поставленных задач, а именно в решении арифметических выражений, содержащих простейшие арифметические операции и скобки.

**Список литературы**

1. C++ In-Depth Box Set First Edition, Vol. 1, 2: Accelerated C++: Practical Programming by Example. 2015.
2. «Язык Си», Год: 2009 Автор: Керниган Б., Ритчи Д. Издательство: Вильямс
3. Онлайн справочник программиста на C и C++ [Электронный курс]. Доступ через сеть Internet по адресу http://www.c-cpp.ru/
4. Программирование [Электронный курс]. Доступ через сеть Internet по адресу hhttp://learnprogramm.ucoz.ru/

**Приложение**

Проект компилировался в CodeBlocks, язык программирования C.

Всего строк в коде программы: 340.

Реализовано структур: 3, реализовано функций: 15.

|  |
| --- |
| 1. #include <stdio.h> 2. #include <math.h> 3. #include <string.h> 4. #include <locale.h> 6. **typedef** **struct** stack3 { 7. **double** bkvnum; 8. **char** thisbkv; 9. **struct** stack3 \*next; 10. }stackbkv; 12. **typedef** **struct** stack2 { 13. **char** thisznak; 14. **struct** stack2 \*next; 15. } stackchar; 17. **typedef** **struct** stack1 { 18. **double** thisnum; 19. **struct** stack1 \*next; 20. } stack; 22. **double** peak(**const** stack \*s) { 23. **return** s->thisnum; 24. } 26. **char** peakchar(**const** stackchar \*s) { 27. **if** (!s) { 28. **return** -1; 29. } 30. **return** priora(s->thisznak); 31. } 33. **int** getsize(stack \*s) { 34. **int** size = 0; 35. **while** (s) { 36. size++; 37. s = s->next; 38. } 39. **return** size; 40. } 42. **double** found(stackbkv \*s, **char** c) { 43. **while** (s) { 44. **if** (c == s->thisbkv) { 45. **return** (s->bkvnum); 46. } 47. s = s->next; 48. } 49. **return**(-1); 50. } 52. **char** popbkv(stackbkv \*\*s) { 53. stackbkv \*r; 54. r = \*s; 55. **char** i = r->thisbkv; 56. \*s = (\*s)->next; 57. free(r); 58. **return** i; 59. } 61. **char** popchar(stackchar \*\*s) { 62. stackchar \*r; 63. r = \*s; 64. **char** i = r->thisznak; 65. \*s = (\*s)->next; 66. free(r); 67. **return** i; 68. } 70. **void** pushbkv(stackbkv \*\*s, **char** item, **double** item1) { 71. stackbkv \*r = malloc(**sizeof**(stackbkv)); 72. r->thisbkv = item; 73. r->bkvnum = item1; 74. r->next = \*s; 75. \*s = r; 76. } 78. **void** pushchar(stackchar \*\*s, **char** item) { 79. stackchar \*r = malloc(**sizeof**(stackchar)); 80. r->thisznak = item; 81. r->next = \*s; 82. \*s = r; 83. } 85. **double** popnum(stack \*\*s) { //снимает 86. stack \*r; 87. r = \*s; 88. **double** i = r->thisnum; 89. \*s = (\*s)->next; 90. free(r); 91. **return** i; 92. } 94. **void** pushnum(stack \*\*s, **double** item) { //кладёт 95. stack \*r = malloc(**sizeof**(stack)); 96. r->thisnum = item; 97. r->next = \*s; 98. \*s = r; 99. } 101. **int** priora(**int**); //приоритет 102. **void** Input(**void**); 103. **void** memath(**void**); 104. **int** oper(**char**); 106. stack \*s = NULL; 107. stackchar \*s2 = NULL; 108. stackbkv \*s3 = NULL; 109. **char** oneChar = 0, stackChar = 0; 110. **double** kek1 = 0, kek2 = 0; //число и знак (1 и 2) 111. **char** c = 0, d = 0; 112. **int** decel = 0; //режим работы 113. **int** zpt = 0; 114. **double** bkv4 = 0; 115. **char** bkv = 0; 116. **int** skobo4ka0 = 0; 117. **char** matan[47]; 118. **char** C = 0; 120. **int** main() { 121. setlocale(LC\_ALL, "russian\_russia.1251"); 122. Input(); 123. printf("%fotv\n", peak(s)); 124. getchar(); 125. } 127. **void** Input() { 128. printf("Введите сначала пременные и их значения, в конце поставьте ';'  (A = 5);\n"); 129. printf("Если таких нет, то поставьте просто ';'\n"); 130. **while** (c != '\t') { 131. d = getchar(); 132. **while** (d != '\n') { 133. **if** (((d >= 'A') && (d <= 'Z')) || ((d >= 'a') && (d <= 'z'))) { 134. bkv = d; 135. } 136. **if** ((kek2 == 1) && (d >= '0') && (d <= '9') && (decel == 0)) { 137. bkv4 = 10 \* bkv4 + d - '0'; 138. } 139. **if** ((d >= '0') && (d <= '9') && (decel == 1) && (kek2 == 1)) { 140. zpt++; 141. bkv4 = bkv4 + (d - '0')\*pow(10, -zpt); 142. } 143. **if** (d == '=') { 144. kek2 = 1; 145. } 146. **if** (d == ';') { 147. pushbkv(&s3, bkv, bkv4); 148. bkv4 = 0; 149. decel = 0; 150. kek2 = 0; 151. } 152. d = getchar(); 153. } 154. printf("Введите вашу строку и нажмите 'ENTER'\n"); 155. c = getchar(); 156. **while** (1) { 157. **int** z = 0; 158. **if** (((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z'))) { 159. matan[0] = c; 160. c = getchar(); 161. **while** (((c >= 'A') && (c <= 'Z')) || ((c >= 'a') && (c <= 'z'))) { 162. z++; 163. matan[z] = c; 164. c = getchar(); 165. } 166. z++; 167. matan[z] = '\0'; 168. **if** (strcmp(matan, "sin") == 0) { 169. C = 's'; 170. } 171. **else** **if** (strcmp(matan, "cos") == 0) { 172. C = 'c'; 173. } 174. **else** **if** (strcmp(matan, "tg") == 0) { 175. C = 't'; 176. } 177. **else** **if** (strcmp(matan, "ctg") == 0) { 178. C = 'g'; 179. } 180. **else** **if** (strcmp(matan, "ln") == 0) { 181. C = 'l'; 182. } 183. **else** { 184. **if** (matan[1] == '\0') { 185. printf("Корректно!\n"); 186. kek1 = found(s3, matan[0]); 187. } 188. **else** { 189. printf("Не корректное выражение!\n"); 190. **return** ; 191. } 192. } 193. } 194. **if** ((c >= '0') && (c <= '9') && (decel == 0)) { 195. kek1 = 10 \* kek1 + c - '0'; 196. } 197. **if** ((c >= '0') && (c <= '9') && (decel == 1)) { 198. zpt++; 199. kek1 = kek1 + (c - '0')\*pow(10, -zpt); 200. } 201. **if** (c == '.') { 202. decel = 1; 203. } 204. **if** (oper(c)) { 205. pushnum(&s, kek1); 206. **if** ((getsize(s) - skobo4ka0) >= 2) { 207. **if** (priora(c) <= priora(peakchar(s2))) { 208. stackChar = popchar(&s2); 209. memath(); 210. } 211. **if** (c != '\n') { 212. pushchar(&s2, c); 213. } 214. } 215. **else** { 216. pushchar(&s2, c); 217. } 218. decel = 0; 219. kek1 = 0; 220. } 221. **if** (c == '\n') { 222. pushnum(&s, kek1); 223. **while** (s2) { 224. stackChar = popchar(&s2); 225. printf("/n%c\n", stackChar); 226. memath(); 227. } 228. **break**; 229. } 230. **if** (c == '(') { 231. pushchar(&s2, c); 232. **if** (C != 0) { 233. pushchar(&s2, C); 234. C = 0; 235. } 236. skobo4ka0 = getsize(s); 237. } 238. **if** (c == ')') { 239. skobo4ka0 = 0; 240. stackChar = popchar(&s2); 241. pushnum(&s, kek1); 242. printf("%fkekus\n", peak(s)); 243. decel = 0; 244. kek1 = 0; 245. **while** (stackChar != '(') { 246. memath(); 247. printf("%f(Временно это ответ)\n", peak(s)); 248. stackChar = popchar(&s2); 249. } 250. kek1 = peak(s); 251. } 252. c = getchar(); 253. } 254. c = getchar(); 255. } 256. } 258. **void** memath() { 259. **double** a = popnum(&s), b=0; 260. **switch** (stackChar) { 261. **case** '+': 262. b = popnum(&s); 263. b = a + b; 264. printf("%f", b); 265. pushnum(&s, b); 266. **break**; 267. **case** '-': 268. b = popnum(&s); 269. b = b - a; 270. printf("%f", b); 271. pushnum(&s, b); 272. **break**; 273. **case** '\*': 274. b = popnum(&s); 275. b = a \* b; 276. printf("%f", b); 277. pushnum(&s, b); 278. **break**; 279. **case** '/': 280. b = popnum(&s); 281. b = b / a; 282. printf("%f", b); 283. pushnum(&s, b); 284. **break**; 285. **case** '^': 286. b = popnum(&s); 287. b = pow(b, a); 288. printf("%f", b); 289. pushnum(&s, b); 290. **break**; 291. **case**'s': 292. b = sin(a); 293. pushnum(&s, b); 294. **break**; 295. **case**'c': 296. b = cos(a); 297. pushnum(&s, b); 298. **break**; 299. **case**'t': 300. b = tan(a); 301. pushnum(&s, b); 302. **break**; 303. **case**'g': 304. b = 1/tan(a); 305. pushnum(&s, b); 306. **break**; 307. **case**'l': 308. b = log(a); 309. pushnum(&s, b); 310. **break**; 311. **default**: 312. ; 313. } 314. } 316. **int** priora(**int** c) { 317. **switch** (c) { 318. **case** '+': 319. **return** 1; 320. **case** '-': 321. **return** 1; 322. **case** '\*': 323. **return** 2; 324. **case** '/': 325. **return** 2; 326. **case** '^': 327. **return** 2; 328. **case** '\n': 329. **return** 0; 330. **default**: 331. ; 332. } 333. } 335. **int** oper(**char** c) { 336. **if** ((c == '/') || (c == '+') || (c == '-') || (c == '\*') || (c == '^') /\*|| (c == '\n')\*/) { 337. **return** 1; 338. } 339. **return** 0; 340. } |

Файл «main.c»