|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 7** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема:**  **«Применение стека и очереди при преобразовании и вычислении арифметических выражений»** | |
|  | |
| Выполнил студент группы ИКБО-07-22 | Козлов К.И. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

# **Цель работы**

# Получение знаний и практических навыков по работе с различными нотациями арифметических выражений.

# **Постановка задачи**

|  |  |
| --- | --- |
| Упражнение 1 | 1. Провести преобразование инфиксной записи выражения в префиксную нотацию, расписывая процесс по шагам   S=a+(b-c\*k)-d\*e-f   1. Представить постфиксную нотацию выражений   a+(c-b)/(b\*d)  (a+b)\*c-(d+e\*f/((g/h+i-j)\*k))/r   1. Представить префиксную нотацию выражений п.2 2. Провести вычисление значения выражения, представленного в постфиксной форме, расписывая процесс по шагам:   7 2 – 3 2 + / 3 + 4 \* |
| Упражнение 2 | 1. Реализовать класс стек, реализующий структуру и методы: втолкнуть элемент в стек, вытолкнуть элемент из стека, вернуть значение элемента в вершине стека, сделать стек пустым, определить пуст ли стек. Рассмотреть два варианта реализации: на массиве; на однонаправленном списке. Интерфейс программы должен обеспечивать непрерывную работу со структурой. 2. Разработать программу сложения двух больших целых чисел (не попадающих в диапазон стандартных типов), вводимых с клавиатуры, как последовательность символов. |

# **Решение**

Стек – это линейный список, в котором операции доступа, вставки, удаления выполняются в вершине списка, т.е. над элементом первой позиции. Это устройство стека называется Last Input First Output (LIFO). Преимуществом стека является константное время выполнение операций вставки и удаления (шаблонный стек).

Очередь – линейный список, в котором операция вставки выполняется в конец списка, а операции удаление и доступ к элементу в вершине. Это устройство очереди называется First Input First Output.

Задание 1

S = a + (b – c\* k) – d\* e – f;

Читаем эту строку с конца. Каждый элемент считываем. Если это операнд, то в строку. Если это оператор, то в стек.

|  |  |  |
| --- | --- | --- |
| Элемент выражения | Строка | Стек |
| f | f |  |
| - | fe | - |
| e | fe | - |
| \* | fe | -\* |
| d | fed | -\* |
| - | fed\*- | - |
| ( | fed\*- | -( |
| k | fed\*-k | -( |
| \* | fed\*-k | -(\* |
| c | fed\*-kc | -(\* |
| - | fed\*-kc\* | -(- |
| b | fed\*kc\*b | -(- |
| ) | fed\*kc\*b- | - |
| + | fed\*kc\*b- - | + |
| a | fed\*kc\*b - - a + |  |

Считаем полученную строку слева направо

Ans = + a - - b \* ck - \* def;

Задание 2 (1)

S = a + (c – b) / (b \* d);

S = a (c- b) / (b \* d) +

S = acb – (b \* d) / +

S = acb – bd \* / +

Задание 2 (2)

S = (a + b) \* c - ( d + e \* f / (( g / h + i – j ) \* k ) ) / r

S = ab + c \* ( d + e \* f / (( g / h + i – j ) \* k ) ) / r -

S = ab + c \* d ( e \* f / (( g / h + i – j ) \* k ) /r) + -

S = ab + c \* d (ef \* f ((g / h + i – j ) \* k ) / r ) / + -

Ans = ab + c \* def \* gh / i + j – k \* + r / -

Задание 3 (1)

S = a + (c – b) / (b \* d);

S = + a (c - b) / (b \* d);

S = + a / (c – b) (b \* d);

Ans = + a / - cd \* bd;

Задание 3 (2)

S = (a + b) \* c - ( d + e \* f / (( g / h + i – j ) \* k ) ) / r;

S = - (a + b) \* c (d + e \* f / ((g / h + i – j) \* k) ) / r;

Ans = - \* + abc / + d \* e / f\* + / gh – i j k r;

Задание 4

|  |  |  |
| --- | --- | --- |
| Элемент выражения | Строка | Стэк |
| 7 | 72 – 32 + / 3 + 4 \* |  |
| 2 | 2 – 32 + / 3 + 4 \* | 7 |
| - | - 32 + / 3 + 4 \* | 72 |
| 3 | 32 + / 3 + 4 \* | 72 - = 5 |
| 2 | 2 + / 3 + 4 \* | 5 3 |
| + | + / 3 + 4 \* | 5 32 |
| / | / 3 + 4 \* | 5 32 + = 5 5 |
| 3 | 3 + 4 \* | 5 5 / = 1 |
| + | + 4 \* | 1 3 |
| 4 | 4 \* | 1 3 + = 4 |
| \* | \* | 4 4 |
|  |  | 4 4 \* = 16 |

Ans = 16

В ходе решения программных задач были написаны некоторые структуры, функции и методы.

Структура node представляет собой узел в списке (стеке)

|  |
| --- |
| struct node  {  int data;  node\* nextNodePtr;  node(int userData)  {  data = userData;  nextNodePtr = nullptr;  }  }; |

Структура stack реализует работу стека как односвязного списка. Имеет методы с функционалом втолкнуть элемент, вытолкнуть элемент, отчистить, проверить пустой ли стек, вернуть значение в вершине списка.

|  |
| --- |
| struct stack  {  node\* tail;  stack()  {  tail = nullptr;  }  bool Empty()  {  if (tail == nullptr)  {  return (true);  }  else  {  return (false);  }  }  void PushBack(int userdata)  {  node\* app = new node(userdata);  if (Empty())  {  tail = app;  }  else  {  app->nextNodePtr = tail;  tail = app;  }  }  void ShowList()  {  if (Empty())  {  return;  }  else  {  node\* ptr = tail;  while (ptr != nullptr)  {  std::cout << ptr->data << ' ';  ptr = ptr->nextNodePtr;  }  }  }  void PopBack()  {  node\* deleteMark = tail;  tail = tail->nextNodePtr;  delete deleteMark;  }  void ReturnTail()  {  std::cout << tail->data;  }  void MakeEmpty()  {  while (tail != nullptr)  {  PopBack();  }  }  int ReturnPop()  {  return (tail->data);  }  }; |

# Для реализации стека на основе массива были написаны следующие функции: add(), pop(), returnTail(), clear(), isClear(), show().

# Функция add() имеет в качестве входных параметров указатель на массив steck, указатель на переменную, отвечающую за длину i и переменную пользовательского ввода. Функция реализует метод pushback в стеке.

|  |
| --- |
| void add(int\* steck, int\* i, int input)  {  steck[\*i + 1] = input;  (\*i)++;  } |

# Функция pop() имеет в качестве входных параметров указатель на массив steck, указатель на переменную, отвечающую за длину i. Функция реализует метод popback в стеке.

|  |
| --- |
| void pop(int\* steck, int\* i)  {  if (\*i == -1) return;  (\*i)--;  } |

# Функция returnTail() имеет в качестве входных параметров указатель на массив steck, переменную, отвечающую за длину i. Функция реализует метод returnTail в стеке.

|  |
| --- |
| int returnTail(int\* steck, int i)  {  return (steck[i]);  } |

Функция clear() имеет в качестве входных параметров указатель на массив steck, переменную, отвечающую за длину i. Функция реализует метод makeEmpty в стеке.

|  |
| --- |
| void clear(int\* steck, int\* i)  {  \*i = -1;  } |

Функция isClear() имеет в качестве входных параметров переменную, отвечающую за длину i. Функция реализует метод isclear в стеке.

|  |
| --- |
| bool isClear(int i)  {  if (i == -1) return (true);  return(false);  } |

Функция show() имеет в качестве входных параметров указатель на массив steck, переменную, отвечающую за длину i.Функция реализует метод showlist в стеке.

|  |
| --- |
| void show(int\* steck, int i)  {  if (isClear(i)) return;  while (i != -1)  {  std::cout << returnTail(steck, i) << ' ';  i--;  }  std::cout << '\n';  } |

Для реализации сложения двух больших чисел были разработаны две функции: convert, sumnums.

Функция convert() имеет в качестве входных параметров строковую переменную число, введенное пользователем. Функция преобразует строку в стек, чтоб было возможно работать с ним далее

|  |
| --- |
| stack\* Convert(std::string num)  {  stack\* big = new stack;  for (int i = 0; i < num.length(); i++)  {  big->PushBack(num[i] - '0');  }  return (big);  } |

Функция sumnums() имеет в качестве входных параметров два стека – два числа, которые будут складываться. Функция реализует сумму двух чисел, даже тех, что не попадают в стандартные типы.

|  |
| --- |
| stack\* SumNums(stack\* a, stack\* b)  {  stack\* sum = new stack;  int overTen = 0;  while (a->tail != nullptr && b->tail != nullptr)  {  sum->PushBack((a->ReturnPop() + b->ReturnPop() + overTen) % 10);  overTen = (a->ReturnPop() + b->ReturnPop() + overTen) / 10;  a->PopBack();  b->PopBack();  }  if (a->tail != nullptr || b->tail != nullptr)  {  if (b->tail != nullptr)  {  a = b;  }  while (a->tail != nullptr)  {  sum->PushBack((a->ReturnPop() + overTen) % 10);  overTen = (a->ReturnPop() + overTen) / 10;  a->PopBack();  }  }  if (overTen != 0)  {  sum->PushBack(overTen);  }  return (sum);  } |

# **4. Тестирование**

При запуске программы пользователь видит интерфейс.

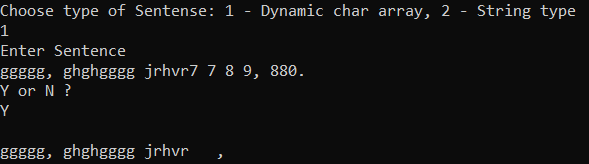


Рисунок 1. Интерфейс программы

Протестируем работу программы со стеком, реализованным на односвязном списке. После выбора режима работы добавим число, удалим это число (push, pop), затем вернем число в вершине, опустошим стек и проверим на пустоту

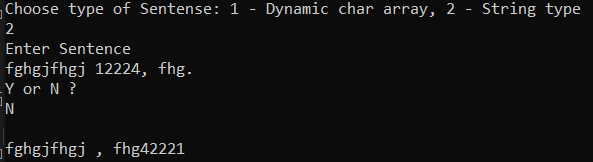


Рисунок 2. Реализация стека на односвязном списке

Далее протестируем работу программы со стеком, реализованным на массиве. Введем все команды в таком же порядке

Рисунок 3. Реализация стека на массиве

Последнее, проверим работу функций, которые реализуют сложение больших чисел. Для этого введем два числа и проверим результат.

Рисунок 4. Проверка суммы двух больших чисел

# **5. Выводы**

В результате выполнения работы я:

1. Освоены навыки в разработке алгоритмов обработки текста – извлечение отдельных элементов
2. Освоены навыки использования средств языка Си и С++ для реализации алгоритмов обработки текстовых данных.

# **6. Исходный код программы**

|  |
| --- |
| #include <iostream>  //////////////////////////////  struct node  {  int data;  node\* nextNodePtr;  node(int userData)  {  data = userData;  nextNodePtr = nullptr;  }  };  struct stack  {  node\* tail;  stack()  {  tail = nullptr;  }  bool Empty()  {  if (tail == nullptr)  {  return (true);  }  else  {  return (false);  }  }  void PushBack(int userdata)  {  node\* app = new node(userdata);  if (Empty())  {  tail = app;  }  else  {  app->nextNodePtr = tail;  tail = app;  }  }  void ShowList()  {  if (Empty())  {  return;  }  else  {  node\* ptr = tail;  while (ptr != nullptr)  {  std::cout << ptr->data << ' ';  ptr = ptr->nextNodePtr;  }  }  }  void PopBack()  {  node\* deleteMark = tail;  tail = tail->nextNodePtr;  delete deleteMark;  }  void ReturnTail()  {  std::cout << tail->data;  }  void MakeEmpty()  {  while (tail != nullptr)  {  PopBack();  }  }  int ReturnPop()  {  return (tail->data);  }  };  /////////////////////////////////////////////////  stack\* Convert(std::string num)  {  stack\* big = new stack;  for (int i = 0; i < num.length(); i++)  {  big->PushBack(num[i] - '0');  }  return (big);  }  stack\* SumNums(stack\* a, stack\* b)  {  stack\* sum = new stack;  int overTen = 0;  while (a->tail != nullptr && b->tail != nullptr)  {  sum->PushBack((a->ReturnPop() + b->ReturnPop() + overTen) % 10);  overTen = (a->ReturnPop() + b->ReturnPop() + overTen) / 10;  a->PopBack();  b->PopBack();  }  if (a->tail != nullptr || b->tail != nullptr)  {  if (b->tail != nullptr)  {  a = b;  }  while (a->tail != nullptr)  {  sum->PushBack((a->ReturnPop() + overTen) % 10);  overTen = (a->ReturnPop() + overTen) / 10;  a->PopBack();  }  }  if (overTen != 0)  {  sum->PushBack(overTen);  }  return (sum);  }  /////////////////////////////////////////////////  void add(int\* steck, int\* i, int input)  {  steck[\*i + 1] = input;  (\*i)++;  }  void pop(int\* steck, int\* i)  {  if (\*i == -1) return;  (\*i)--;  }  int returnTail(int\* steck, int i)  {  return (steck[i]);  }  void clear(int\* steck, int\* i)  {  \*i = -1;  }  bool isClear(int i)  {  if (i == -1) return (true);  return(false);  }  void show(int\* steck, int i)  {  if (isClear(i)) return;  while (i != -1)  {  std::cout << returnTail(steck, i) << ' ';  i--;  }  std::cout << '\n';  }  ////////////////////////////////////////////////  int main()  {  int control;  int i = -1;  int k;  int\* steck = new int[10];  do  {  std::cout << "What u want ?" << std::endl;  std::cout << "1 - list Stack" << std::endl;  std::cout << "2 - mass Stack" << std::endl;  std::cout << "3 - sum Stack" << std::endl;  std::cin >> control;  switch (control)  {  case 1:  {  stack user;  int controlOne;  do  {  std::cout << "What u want ?" << std::endl;  std::cout << "1 - Push" << std::endl;  std::cout << "2 - Pop" << std::endl;  std::cout << "3 - clear" << std::endl;  std::cout << "4 - clear check" << std::endl;  std::cout << "5 - return head" << std::endl;  std::cin >> controlOne;  switch (controlOne)  {  case 1:  {  int userPush;  std::cout << "Enter node data" << std::endl;  std::cin >> userPush;  user.PushBack(userPush);  user.ShowList();  std::cout << std::endl;  break;  }  case 2:  {  user.PopBack();  user.ShowList();  std::cout << std::endl;  break;  }  case 3:  {  user.MakeEmpty();  std::cout << std::endl;  break;  }  case 4:  {  user.Empty();  std::cout << std::endl;  break;  }  case 5:  {  user.ReturnTail();  std::cout << std::endl;  break;  }  }  } while (controlOne == 1 || controlOne == 2 || controlOne == 3 || controlOne == 4 || controlOne == 5);  break;  }  case 2:  {  int input;  std::cout << "Enter start amount of nums in stack" << std::endl;  std::cin >> k;  std::cout << "Enter start nums in stack" << std::endl;  for (int j = 0; j < k; j++)  {  std::cin >> input;  add(steck, &i, input);  }  int controlOne;  do  {  std::cout << "What u want ?" << std::endl;  std::cout << "1 - Push" << std::endl;  std::cout << "2 - Pop" << std::endl;  std::cout << "3 - clear" << std::endl;  std::cout << "4 - clear check" << std::endl;  std::cout << "5 - return head" << std::endl;  std::cin >> controlOne;  switch (controlOne)  {  case 1:  {  int userPush;  std::cout << "Enter node data" << std::endl;  std::cin >> userPush;  add(steck, &i, userPush);  show(steck, i);  std::cout << std::endl;  break;  }  case 2:  {  pop(steck, &i);  show(steck, i);  std::cout << std::endl;  break;  }  case 3:  {  clear(steck, &i);  std::cout << std::endl;  break;  }  case 4:  {  isClear(i);  std::cout << std::endl;  break;  }  case 5:  {  returnTail(steck, i);  std::cout << std::endl;  break;  }  }  } while (controlOne == 1 || controlOne == 2 || controlOne == 3 || controlOne == 4 || controlOne == 5);  break;  }  case 3:  {  std::string num1;  std::string num2;  std::cout << "Enter Two nums" << std::endl;  std::cin >> num1;  std::cin >> num2;  stack\* res = SumNums(Convert(num1), Convert(num2));  std::cout << "Here it is !" << std::endl;  res->ShowList();  break;  }  }  } while (control == 1 || control == 2 || control == 3);  } |