**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Моделирование стека

|  |  |  |
| --- | --- | --- |
| Студент гр. 6303 |  | Жахин А.А. |
| Преподаватель |  | Берленко Т.А. |

Санкт-Петербург

2017

**Цель работы:** Требуется написать программу, моделирующую работу стека, реализовав перечисленные ниже методы. Программе на вход подается последовательность команд с новой строки (не более 100 команд), в зависимости от которых программа выполняет ту или иную операцию и выводит результат ее выполнения с новой строки.  
  
Перечень команд:

* **push n** -  добавляет целое число n в стек. Программа должна вывести **"ok"**
* **pop** - удаляет из стека последний элемент и выводит его значение на экран
* **top** - программа должна вывести верхний элемент стека на экран не удаляя его из стека
* **size** - программа должна вывести количество элементов в стеке
* **exit** - программа должна вывести "**bye**" и завершить работу

Если в процессе вычисления возникает ошибка (например вызов метода **pop** при пустом стеке), программа должна вывести "**error**" и завершиться.

Стек требуется реализовать самостоятельно на базе **массива**.

**Ход работы:**

1. Создаются функции для управления стеком:

А) **push n**

|  |
| --- |
| void pushN(int \*arr, int \*sizeSt, int n)  {  arr[\*sizeSt] = n;  \*sizeSt = \*sizeSt+1;  printf("ok\n");  } |

Б) **pop**

|  |
| --- |
| int pop(int \*arr, int \*sizeSt)  {  if(\*sizeSt != 0)  {  printf("%d\n", arr[\*sizeSt-1]);  arr[\*sizeSt-1] = 0;  \*sizeSt = \*sizeSt - 1;  return 0;  }  else  {  printf("error");  return 1;  }  } |

В) **top**

|  |
| --- |
| int top(int \*arr, int \*sizeSt)  {  if(\*sizeSt != 0)  {  printf("%d\n", arr[\*sizeSt-1]);  return 0;  }  else  {  printf("error");  return 1;  }  } |

Г) **size**

|  |
| --- |
| void size(int \*sizeSt)  {  printf("%d\n", \*sizeSt);  } |

Д) **exit (runaway)**

|  |
| --- |
| void runAway()  {  printf("bye");  } |

Далее в функции **main()** создается массив типа **char**, в который будут поступать команды пользователя.

|  |
| --- |
| int main()  {  int array[100];  int n;  char command[100];  char \*arrayPtr[2]; //массив указателей на строки, в которых будут храниться слова из текста  int j;  int sizeSt = 0; |

После того, как будет считана команда, с помощью функции **strtok()** команда делится на отдельные слова, и если второе слово существует, оно преобразуется в тип **int** и присваивается переменной **n**.

|  |
| --- |
| fgets(command, 100, stdin);  arrayPtr[0] = strtok(command, " "); // нулевому указателю присваеваеncz первое слово  arrayPtr[1] = strtok(NULL, " "); //второе слово  if(arrayPtr[1] != NULL)  {  n = atoi(arrayPtr[1]);  } |

Первое слово передается функции **cmp**:

|  |
| --- |
| int cmp(char \*a)  {  if (strcmp (a, "push")==0)  return 1;  if (strcmp (a, "pop\n")==0)  return 2;  if (strcmp (a, "top\n")==0)  return 3;  if (strcmp (a, "size\n")==0)  return 4;  if (strcmp (a, "exit\n")==0)  return 5;  else  return 0;  return 0;  } |

Далее, в зависимости от возвращаемого значения функции **cmp**, выполняется введенная пользователем команда. Затем командные строки обнуляются и на вход принимается новая команда.

|  |
| --- |
| int var = cmp(arrayPtr[0]);  switch (var)  {  case 1:  {  pushN(array, &sizeSt, n);  break;  }  case 2:  {  if(pop(array, &sizeSt))  return 0;  break;  }  case 3:  {  if(top(array, &sizeSt))  return 0;  break;  }  case 4:  {  size(&sizeSt);  break;  }  case 5:  {  runAway();  return 0;  break;  }  case 0:  {  printf("error");  return 0;  break;  }  }  arrayPtr[0] = NULL;  arrayPtr[1] = NULL; |

Исходный код:

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  void pushN(int \*arr, int \*sizeSt, int n)  {  arr[\*sizeSt] = n;  \*sizeSt = \*sizeSt+1;  printf("ok\n");  }  int pop(int \*arr, int \*sizeSt)  {  if(\*sizeSt != 0)  {  printf("%d\n", arr[\*sizeSt-1]);  arr[\*sizeSt-1] = 0;  \*sizeSt = \*sizeSt - 1;  return 0;  }  else  {  printf("error");  return 1;  }  }  int top(int \*arr, int \*sizeSt)  {  if(\*sizeSt != 0)  {  printf("%d\n", arr[\*sizeSt-1]);  return 0;  }  else  {  printf("error");  return 1;  }  }  void size(int \*sizeSt)  {  printf("%d\n", \*sizeSt);  }  void runAway()  {  printf("bye");  }  int cmp(char \*a)  {  if (strcmp (a, "push")==0)  return 1;  if (strcmp (a, "pop\n")==0)  return 2;  if (strcmp (a, "top\n")==0)  return 3;  if (strcmp (a, "size\n")==0)  return 4;  if (strcmp (a, "exit\n")==0)  return 5;  else  return 0;  return 0;  }  int main()  {  int array[100];  int n;  char command[100];  char \*arrayPtr[2]; //массив указателей на строки, в которых будут храниться слова из текста  int j;  int sizeSt = 0;  for(j=0; j<100; j++)  {  fgets(command, 100, stdin);  arrayPtr[0] = strtok(command, " "); // нулевому указателю присваеваеncz первое слово  arrayPtr[1] = strtok(NULL, " "); //второе слово  if(arrayPtr[1] != NULL)  {  n = atoi(arrayPtr[1]);  }  int var = cmp(arrayPtr[0]);  switch (var)  {  case 1:  {  pushN(array, &sizeSt, n);  break;  }  case 2:  {  if(pop(array, &sizeSt))  return 0;  break;  }  case 3:  {  if(top(array, &sizeSt))  return 0;  break;  }  case 4:  {  size(&sizeSt);  break;  }  case 5:  {  runAway();  return 0;  break;  }  case 0:  {  printf("error");  return 0;  break;  }  }  arrayPtr[0] = NULL;  arrayPtr[1] = NULL;  }  return 0;  } |

Вывод: в ходе лабораторной работы была написана программа, моделирующая работу стека. Стек был реализован на базе массива.