**Сравнительный анализ характеристик реализаций структур хранения данных «стек» и «список» в различных языках программирования**

Цель работы: сравнить стандартную реализацию структур хранения данных, и реализацию структур, представленных в виде массива в виде односвязного линейного списка

**Стек**

**Реализация стека в виде массива.** Если стек реализован в виде статического или динамического массива (вектора), то для его хранения обычно отводится непрерывная область памяти ограниченного размера, имеющая нижнюю и верхнюю границу. Перед началом работы указатель стека находится ниже левой (нижней) границы массива. При включении первого элемента в стек указатель устанавливается на начало массива и по адресу первого элемента размещается записываемое значение. При попытке добавить в стек каждый последующий элемент сначала происходит смещение указателя на длину типа данных (т.е. он перемещается к следующему элементу массива), а затем – размещение значений элементов по этим адресам. Если указатель выходит за верхнюю границу массива, то это признак того, что стек переполнен. При исключении элемента из стека сначала считываются данные, а затем происходит перемещение указателя к предыдущему элементу. Если указатель стека выходит за нижнюю границу массива, то стек пуст.

**Реализация стека в виде списка.** До начала работы указатель стека показывает на нулевой, физически отсутствующий адрес (т. е. указатель - пустой). При включенииэлемента в стек сначала происходит выделение области памяти, адрес которой записывается в указатель стека, а затем по значению этого указателя в стек помещается информация.

При исключенииэлемента сначала по указателю стека считывается информация об исключаемом элементе, а затем указатель смещается к предыдущему элементу. После чего освобождается память, выделенная под элемент. Если указатель имеет значение нулевого адреса, то стек пуст.

**С++**

**Внутренняя структура данных**

**Для реализации списком**

**struct** Node {

**int** value;

**struct** Node \***next**;

} ;

**Node** \*list=0; - указатель вершин стека

**value –** слово.

**\*next** – указатель на следующий элемент списка.

**Для реализации массивом**

**int** \*p – указатель вершины стека

**int**  b[10] массив для хранения значений

**Функции работы со стеком, реализация- список.**

**Извлечение элемента:**

**template** < **typename** T>

**void** pop2(T\*\* head) {

T \*out;

**if** (\*head == **NULL**) {

exit(-1);

}

out = \*head;

\*head = (\*head)->next;

free(out);

**return** ;

}

**Добавление элемента:**

**void** push(Node \*\*head,**const** **char** val[]) {

Node \*tmp =(Node \*) malloc(**sizeof**(Node));

**if** (tmp == **NULL**) { exit(-1) ; }

tmp->**next** = \*head;

strcpy(tmp->**value**,val);

\*head = tmp;

}

**Функции работы со стеком, реализация- массив**

**Добавление элемента**

**void** add\_array(**char** h[])

{

**if**(p<b[0]) {cout<<**"Stack** **is** **empty** **now"**;p=b[0];}

**if** (p<=(b[9]))

{

strcpy(p,h);

p=p+**sizeof**(b[0]);

}

**else**

{

cout<<**"\nStack** **is** **full** **now"**<<endl;

}

**return**;}

**Удаление элемента**

**void** del\_array()

{

**if** (p>=b[0])

{

p-=**sizeof**(b[0]);

}

**else**

cout<<"\nStack is empty now\n";

**return** ;

}