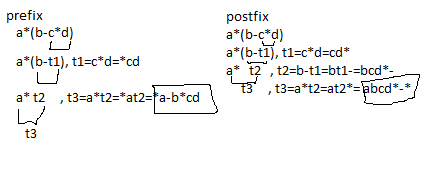
Prefix­­­­­­­­­­­­­­­­­­­­­­­­­\_\_\_\_\_\_\_\_\_\_\_­­­\_\_\_\_\_infix\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_postfix

-ab a-b ab-

a\*(b-c\*d)

prefix based on order of operations

a\*(b-c\*d) //inside the parenthesis



ex1

a=2, b=10, c=3, d=3

a\*(b-c\*d)

a\*(b-t1), t1=c\*d=9

a\* t2, t2=b-t1=10-9=1

t3 , t3=a\*t2=2\*1=2

ex2

\*a-b\*cd //under from post to infix \*cd to c\*d

\*a-b t1 , t1=\*cd=c\*d=3\*3=9

\*a t2 , t2=-bt1=b-t1=10-9=1

t3 , t3 = \*at2=a\*t2=2\*1=2

ex3

abcd\*-\*

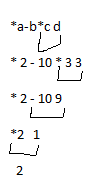
a b t1 - \*, t1=cd\*=c\*d=3\*3=9

a t2 \*, t1=bt1-=b-t1=10-9=1

t3 , t3=at2\*=a\*t2=2\*1=2

ex other

\*a-b\*cd

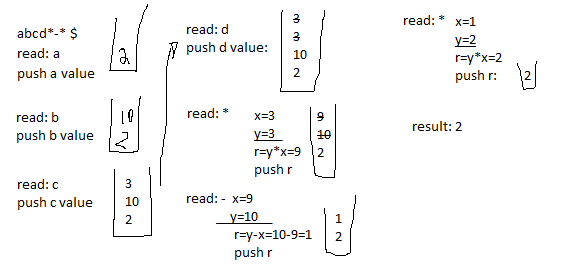


//IN PROGRAM camputor

abcd\*-\* $

read: a

push a value



int a=2,b=10,c=3,d=3;

int x,y,r;

char p;

~~a~~bcd\*-\*$

cin>>p;

while(p!=’$’)

{

switch(p)

{

case ‘a’:s.pushStack(a);

. . .

. . .

. . . //all the other cases through d

case ‘\*’: x=s.popStack();

y=s.popStack();

r=y\*\*;

s.pushStack®;

break;

}

cin>>p;

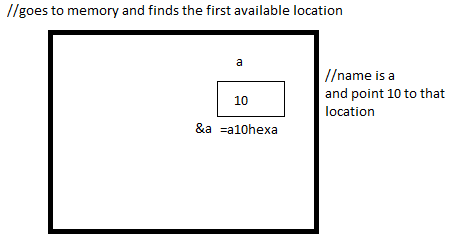
}

NEW PART

PART II

Pointers //uh oh!

int a=10; //what happens when the compiler gets to this statement?



//&a to get the address location of a

cout<<a; //10

cout<<&a; //a10hexa the six digit hexadecimal number

cout<<\*(&a) //the star of any address is the content, so we see 10 again

//the address changes but the content doesn’t

//what if u have a group of those locations

int b[4]={3,1,9,7}; //go to memory and find 4 consecutive locations

b0=b[0], b1=b[1], b2=b[2], b3=b[3] //was bsmall0 or bsub0 bsmall1 etc

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 9 | 7 |

address: {&b[0] &b[1] &b[2] &b[3]

{b+0 b+1 b+2 b+3

cout<<b[2]; //9

cout<<&b[2] //the address

cout<<\*&b[2]; //9

cout<<\*(b+2); //9

int a=10;

a

|  |
| --- |
| 10 |

p=&a

//p is holding the address of a location who’s content is. .

int \*p; //p is a pointer to type integer

read: P”is a pointer to type” int

i.e. P can hold address of a location whose content is int

//now we can make the address of a hold in p

p=&a ;

cout<<a; //10

cout<<\*&a; //10

cout<<\*p; //10

ex.

int a=5, b=10;

|  |
| --- |
| 5 |

|  |
| --- |
| 10 |

&a &b

p=&a q=&b

int \*p;

int \*q;

p=&a; q=&b;

//\*p is 5 and \*q is 10

\*p = \*p+3;

//becomes \*p = 5+3;

|  |
| --- |
| 8 |

\*p=8;// now &a

\*q=b+4;

//becomes \*q=10+4;

|  |
| --- |
| 14 |

\*q=14;// now &b

\*p=\*q;

|  |
| --- |
| 14 |

|  |
| --- |
| 14 |

&a &b

//#########################################

a=10; b=20;

p=&a; q=&b;

a b

|  |
| --- |
| 10 |

|  |
| --- |
| 20 |

p=&a q=&b

exchange the content of a and b

//we will need a temporary location

int temp = a;

a=b;

b=temp; //THIS IS METHOD 1

temp

|  |
| --- |
| 20 |

//METHOD 2

//think of two houses and u want to switch, just switch keys or change addresses

int \*t; //t is a pointer to type integer

|  |
| --- |
| 20 |

a b

|  |
| --- |
| 10 |

p=&a q=&b

t=p; //t is pointing to location of p

p=q; //p is pointing to q or b

q = t; //q is pointing to same location where t was pointing to which was originally p

cout<<\*p<<\*q; //content of p is 20, content of q is 10, we just switched addresses

int b[4]={3,1,9,7};

b[0], b[1], b[2], b[3]

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 9 | 7 |

add: b+0 b+1 b+2 b+3

=b //when u pass array in a function it is by reference

display(b);

void display(int x[ ])

{

for(int i =0; i<4; ++i)

{

cout<<x[i];

}

}//end display

//instead of this we are passing an address

void display(int \*x) //the same?

{

for(int I =0; i<4; ++i)

{

cout<<x[i];

}

}//end display

//we want to display array b;

-for(int i =0; i<4: ++i) cout<<b[i];

-for(i=0; i<4; ++i)

cout<<\*(b+i); //shows u the same bec of their locations

//what will happen if we do cout<<i[b] //apparently does the same job? content of i+b, b+i same thing

-int\*p;

p=b;

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 9 | 7 |

p= b+0 b+1 b+2 b+3

for(i=0;i<4;++i)

{

cout<<\*p; //how can we move p to point to next location?

p++; //is the one moving pointer to next location

}

//now we combine the inside to one statement

cout<<(\*p)++;

//going back to other example

int b[4]={3,1,9,7};

b[0], b[1], b[2], b[3]

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 9 | 7 |

add: b+0 b+1 b+2 b+3

int \*p-b;

cout<<\*p; //\*p points to 3

p++; //p is now pointing to next location bec u added 1 to the next hexadecimal address in the array

cout<<\*p; //\*p points to 1

p=&b[3];

cout<<\*p; //=7

p- -;//p is now pointing to b[2]

cout<<\*p; //=9

p=&b[3];

while(p!=b)

{

cout<<\*p;

p- -;

}

b[0] b[1] b[2] b[3]

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 9 | 7 |

add: b b+1 b+2 b+3

//p right now is pointing to last location //we are comparing 2 addresses

//let’s say b is 100, b+1 is 101, b2 is 102 etc. . .

//so while p is not b keep reducing and output the value of what is stored in p

//output is 7 9 1 //doesn’t print the last bec if it is equal to b 100=100

to print the last one we want to add while(p>=b)

//we are not comparing content, but the addresses.

int a[12]’ //means 12 consecutive locations

//we have the location but let’s say we don’t have them next to each other

memory lets you use as much memory as possible

