**Rules**

1. **Pointer + IC \*DataTypeSIZE**
2. **Pointer - IC \*DataTypeSIZE**
3. **But \* / % No Operation Perform**
4. **1 Pointer copy to Another ex Pointer1 = Pointer2**
5. **Pointer1 – Pointer2 = index minus**
6. **2 Pointer minus not other +\*/%**
7. **Array Name – Base Address**

**Pointers and Arrays**

int main()

{

//i j k

//3 1.5 c

//1002 2002 3002

//x y z

//1002 2002 3002

//4002 5002 6002

int i = 3, \*x ;

float j = 1.5, \*y ;

char k = 'c', \*z ;

printf ( "\nValue of i = %d", i ) ;//3

printf ( "\nValue of j = %f", j ) ;//1.500000

printf ( "\nValue of k = %c", k ) ;//c

x = &i ;

y = &j ;

z = &k ;

printf ( "\nOriginal address in x = %u", x ) ;//1002

printf ( "\nOriginal address in y = %u", y ) ;//2002

printf ( "\nOriginal address in z = %u", z ) ;//3002

//Rule

x=x+4 ;//x = x + 4 =1002+4\*4=1018

y=y+5 ;//y = 2002+5\*4 =2022

z=z+10 ;//z = 3002+10\*1=3012

printf ( "\nNew address in x = %u", x ) ;//1006

printf ( "\nNew address in y = %u", y ) ;//2006

printf ( "\nNew address in z = %u", z ) ;//3003

return 0;

}

As well as –

int main()

{

//i j k

//3 1.5 c

//1002 2002 3002

//x y z

//1002 2002 3002

//4002 5002 6002

int i = 3, \*x ;

float j = 1.5, \*y ;

char k = 'c', \*z ;

printf ( "\nValue of i = %d", i ) ;//3

printf ( "\nValue of j = %f", j ) ;//1.500000

printf ( "\nValue of k = %c", k ) ;//c

x = &i ;

y = &j ;

z = &k ;

printf ( "\nOriginal address in x = %u", x ) ;//1002

printf ( "\nOriginal address in y = %u", y ) ;//2002

printf ( "\nOriginal address in z = %u", z ) ;//3002

//Rule

x=x-4 ;//x = x + 4 =1002+4\*4=1018

y=y-5 ;//y = 2002+5\*4 =2022

z=z-10 ;//z = 3002+10\*1=3012

printf ( "\nNew address in x = %u", x ) ;//1006

printf ( "\nNew address in y = %u", y ) ;//2006

printf ( "\nNew address in z = %u", z ) ;//3003

return 0;

}

1 pointer copy to another pointer

int main()

{

//i j k

//4 1042 1054

//1002 2002 3002

int i = 4, \*j, \*k ;

j = &i ;

printf("%u\n",&i);//1002

printf("%u\n",j);//1002

j = j + 1 ;

printf("%u\n",j);//1006

j = j + 9 ;

printf("%u\n",j);//1042

k = j + 3 ;

printf("%u\n",k);//1042+3\*4=1054

return 0;

}

int main()

{

int arr[ ] = { 10, 20, 30, 45, 67, 56, 74 } ;

int \*i, \*j ;

i = &arr[1] ;

j = &arr[5] ;

printf ( "%d %d", j - i, \*j - \*i ) ;

return 0;

}

//Rule Pointer1 - Pointer2 = Index

//i j

//1006 1022

//2002 3002

//index 0 1 2 3 4 5 6

//value 10 20 30 45 67 56 74

//Address 1002 1006 1010 1014 1018 1022 1026

//j-i = 5 - 1 = 4

//\*j-\*i = \*1022 - \*1006

// 56 - 20 = 36

int main()

{

int arr[ ] = { 10, 20, 36, 72, 45, 36 } ;

int \*j, \*k ;

j = &arr [ 4 ] ;//j=1018

k = ( arr + 4 ) ;//k=1002 + 4\*4=1018 same as k = (4+arr)

if ( j == k )//1018 == 1018

printf ( "The two pointers point to the same location %u %u=> %d %d",j,k,\*j,\*k ) ;

else

printf ( "The two pointers do not point to the same location" ) ;

return 0;

}

//Rule Pointer1 - array name base Address

//i j

//1018 1018

//2002 3002

//index 0 1 2 3 4 5 6

//value 10 20 36 72 45 36 74

//Address 1002 1006 1010 1014 1018 1022 1026

(b) A pointer when **incremented** always points to an **immediately** **next** location of its type.

int main()

{

int arr[ ] = { 10, 20, 36, 72, 45, 36 } ;

int \*ptr;

int n=6,i;

ptr=arr ; //ptr=&arr[0] ptr = arr + 0 =1002 +0\*4 =1002 ptr=0+arr ptr=1002

for(i=0;i<n;i++)//1<6

{

printf("%u => %d\n",ptr,\*ptr);//1006 20

ptr++;//ptr=ptr+1 = 1006+1\*4=1010

}

return 0;

}

//index 0 1 2 3 4 5 6

//value 10 20 36 72 45 36 74

//Address 1002 1006 1010 1014 1018 1022 1026

//j-i = 5 - 1 = 4

//\*j-\*i = \*1022 - \*1006

// 56 - 20 = 36

int main()

{

int arr[ ] = { 10, 20, 36, 72, 45, 36 } ;

int \*ptr;

int n=6,i;

ptr=arr ; //ptr=&arr[0] ptr = arr + 0 =1002 +0\*4 =1002 ptr=0+arr ptr=1002

for(i=0;i<n;i++)//0<6

{

printf("%u => %d %d %d",ptr,\*ptr,arr[i],\*(&arr[i]) );//\*(1006 )

printf(" %d %d %d %d\n",\*(arr+i),\*(i+arr),i[arr],\*(&i[arr]));

ptr++;//ptr=ptr+1 = 1006+1\*4=1010

}

return 0;

}

//index 0 1 2 3 4 5 6

//value 10 20 36 72 45 36 74

//Address 1002 1006 1010 1014 1018 1022 1026

//j-i = 5 - 1 = 4

//\*j-\*i = \*1022 - \*1006

// 56 - 20 = 36

Whole Array Pass To Function

void display(int\*,int);

int main()

{

int arr[ ] = { 10, 20, 36, 72, 45, 36 } ;

int n=6 , i ;

printf("\nIN MAIN\n");

for(i=0;i<n;i++)//1<6

printf("%u=>%d\n",&arr[i],arr[i]);//1006 20

display(arr,n);//display(1002,6);

printf("\nIN MAIN A Function Call\n");

for(i=0;i<n;i++)//1<6

printf("%u=>%d\n",&arr[i],arr[i]);//1006 20

return 0;

}

//ptr

//1006

//2002

void display(int \*ptr,int n)

{

int i;

printf("\nIN FUNCTION\n");

for(i=0;i<n;i++)//1<6

{

printf("%u=>%d\n",ptr,\*ptr);//1002 10

\*ptr=1;

ptr++;

}

}

//index 0 1 2 3 4 5 6

//value 10 20 36 72 45 36 74

//Address 1002 1006 1010 1014 1018 1022 1026

//j-i = 5 - 1 = 4

//\*j-\*i = \*1022 - \*1006

// 56 - 20 = 36