Introduction to C Pointers:

# Pointers:

Pointers are variables that point to the address of another variable.

syntax: type \*name;

E.g. int \*ptr; /\* pointer to integer \*/

# Pointers and Addresses

int x, y;

int \*p, \*q;

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

x y p q ...

x = 5;

y = 10;

p = &x; /\* assign the address of x to p \*/

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **10** | 0x1001 |  |  |

x y p q ...

q = p;

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **5** | **10** | 0x1001 | 0x1001 |  |

x y p q ...

\*q = 6; /\* modify the value q points to \*/

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **6** | **10** | 0x1001 | 0x1001 |  |

x y p q ...

# Important pointer notations

int x; /\* x is an integer \*/

int \*p; /\* declare p as a pointer to integer \*/

p = &x; /\* assign the address of x to p \*/

\*p = 5; /\* modify the memory location p points to \*/

/\* the value of x in now 5 \*/

## Example1:

#include <stdio.h>

int main()

{

int x, y;

int \*p, \*q;

x = 5;

y = 10;

p = &x;

q = &y;

printf( "x = %d\n", x );

printf( "y = %d\n", y );

printf( "p = %p\n", p );

printf( "q = %p\n", q );

/\* modify x & y using pointers \*/

\*p = 10;

\*q = 20;

printf( "x = %d\n", x );

printf( "y = %d\n", y );

/\* using \*p & \*q like int variables \*/

\*p = \*p + 5;

\*q = \*q - 5;

printf( "x = %d\n", x );

printf( "y = %d\n", y );

(\*p)++;

(\*q)--;

printf( "x = %d\n", x );

printf( "y = %d\n", y );

system( "pause" );

return 0;

}

# Pointers and Function Arguments

Function arguments are passed by value in C.

Pointers allow the modification of variables in the calling function by passing the address of variables.

## Example2:

#include <stdio.h>

/\* incorrect version of the swap function \*/

void swap1(int x, int y);

/\* correct version of the swap function \*/

void swap2(int \*p, int \*q);

int main()

{

int x, y;

x = 3;

y = 4;

printf( "main: (x, y) = (%d, %d)\n", x, y );

swap1( x, y );

printf( "main: (x, y) = (%d, %d)\n\n", x, y );

printf( "main: (x, y) = (%d, %d)\n", x, y );

swap2( &x, &y );

printf( "main: (x, y) = (%d, %d)\n\n", x, y );

system( "pause" );

return 0;

}

/\* incorrect implementation of the swap function \*/

void swap1(int x, int y)

{

int temp;

temp = x;

x = y;

y = temp;

printf( "swap1: (x, y) = (%d, %d)\n", x, y );

}

/\* correct implementation of the swap function \*/

void swap2(int \*p, int \*q)

{

int temp;

temp = \*p;

\*p = \*q;

\*q = temp;

printf( "swap2: (x, y) = (%d, %d)\n", \*p, \*q );

# Pointers and Arrays

Pointers and arrays are closely related in C. Any array operation can be done using pointers.

int a[5]; /\* an array of 5 integers \*/

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

The array a:

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

a[0] a[1] a[2] a[3] a[4]

a[i] refers to the i­­­­­th element of the array. p = &a[i]; /\* makes p point to the i­­­­­th element of the array \*/

if p is pointing to &a[0], then &a[i] can also be referred to as (p+i) using pointer notation

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

p p+1 p+2 p+3 p+4

Manipulating arrays using pointers, in general, gives a faster result but code may be less readable for the novice programmer.

## Example3:

#include <stdio.h>

/\* strlen using pointer notation \*/

int strlen1(char str[]);

/\* strlen using array notation \*/

int strlen2(char str[]);

int main(int argc, char \*argv[])

{

printf ( "strlen1(\"hello\") = %d\n", strlen1( "hello" ) );

printf ( "strlen2(\"hello\") = %d\n", strlen2( "hello" ) );

system( "pause" );

return 0;

}

/\* strlen using pointer notation \*/

int strlen1(char str[])

{

int size;

for( size = 0; str[size] != '\0'; size++ )

{

}

return size;

}

/\* strlen using array notation \*/

int strlen2(char str[])

{

int size = 0;

char \*ptr;

for( ptr = &str[0]; \*ptr != '\0'; ptr++ )

{

size ++;

}

return size;

}

# Pointer Arithmetic

If p is a pointer to an array element, then p++ will increment p to point to the next array element, and p+x will make p point to the element x places after the current location p points to

## Example 4

#include <stdio.h>

int main(int argc, char \*argv[])

{

char char\_array[] = "Hello";

int int\_array[] = { 1, 2, 3};

char c1, c2, c3;

int x, y, z;

char \*char\_ptr;

int \*int\_ptr;

char\_ptr = &char\_array[0];

int\_ptr = &int\_array[2];

c1 = \*char\_ptr;

c2 = \*(char\_ptr+1);

c3 = \*(char\_ptr+2);

x = \*int\_ptr;

y = \*(int\_ptr-1);

z = \*(int\_ptr-2);

printf ( "c1 = %c\n", c1 );

printf ( "c2 = %c\n", c2 );

printf ( "c3 = %c\n", c3 );

printf ( "x = %d\n", x );

printf ( "y = %d\n", y );

printf ( "z = %d\n", z );

system( "pause" );

return 0;

}

# Character Pointers

C does not have a built-in string data types. Strings are represented by character arrays

Notations:

char message1[] = “Hello, World!”;

char \*message2 = “hello, World!”;

# Pointers to Pointers

Pointers are variables themselves, so they can be pointed to by other pointers

int x; /\* x is integer \*/

int \*px = &x; /\* px is pointer to integer (x) \*/

int \*\*ppx = &p; /\* ppx is pointer to px \*/

/\* same as saying int \*\*ppx = &&x \*/

0x1001 0x1002 0x1003 0x1004 0x1005

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0x1001 | 0x1002 |  |  |

x px ppx ... ..

# Pointers to Functions

Even though functions are not variables, in C, it is possible to define pointers to functions.

Function pointers, like variables, can be assigned, passed to functions, returned from functions, and be stored in an array.

The format for declaring function pointers looks like the format for declaring function prototypes, but with a pointer notation in front of the name of the function (with in a parenthesis). Look at the example below.

/\* Prototype of a function that takes two integers and returns an integer \*/

int add(int x, int y);

/\* Declaration of a function pointer that takes two integers and returns an integer\*/

int (\*add\_pointer)(int x, int y);

## Example5

#include <stdio.h>

int add(int x, int y);

int (\*add\_pointer)(int, int);

int main()

{

int x = 3;

int y = 4;

add\_pointer = &add;

printf("add x, y is: %d\n", add(x,y));

printf("add\_pointer x, y is: %d\n", add\_pointer(x,y));

}

int add(int a, int b)

{

return a + b;

}