Pointers

Pointers and the Address Operator

- Each variable in a program is stored at a unique address in memory
- Use the address operator **&** to get the address of a variable:

```
int num = -23;
cout << &num; // prints address
              // in hexadecimal
```

- The address of a memory location is a pointer

Pointer Variables

- Pointer variable (pointer): variable that holds an address
- Pointers provide an alternate way to access memory locations & value

Pointer Variables

- Definition:

```
int  *intptr;
```

- Read as:
  "**`intptr`** can hold the address of an int" or "the variable that **`intptr`** points to has type int"

- Spacing in definition does not matter:

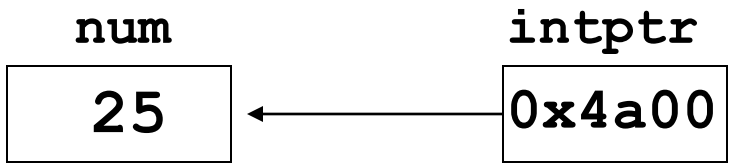
```
int * intptr;
int*  intptr;
```

Pointer Variables

- Assignment:

```
int num = 25;
int *intptr;
intptr = &num;
```

- Memory layout:



address of `num`: `0x4a00`

- Can access **`num`** using **`intptr`** and indirection operator `*`:

```
cout << intptr;  // prints 0x4a00
cout << *intptr; // prints 25
```

Pointer Arithmetic

Some arithmetic operators can be used with pointers:

- Increment and decrement operators **++**, **−−**
- Integers can be added to or subtracted from pointers using the operators **+**, **−**, **+=**, and **−=**
- One pointer can be subtracted from another by using the subtraction operator **−**

Pointer Arithmetic

Assume the variable definitions

```
int vals[]={4,7,11};
int *valptr = vals;
```

Examples of use of `++` and `--`

```
valptr++; // points at 7
valptr--; // now points at 4
```

What is the invalid pointer arithmetic?

One can perform different arithmetic operations on Pointer such as increment,decrement but still we have some more arithmetic operations that cannot be performed on pointer –

1. Addition of two addresses.
2. Multiplying two addresses.
3. Division of two addresses.

1. Addition of Two Pointers :

```
#include<stdio.h>

int main()
{

int var = 10;
int *ptr1 = &i;
int *ptr2 = (int *)2000;

printf("%d",ptr1+ptr2);

return 0;
}
```

Output :

```
Compile Error
```

2. Multiplication of Pointer and number :

```
#include<stdio.h>

int main()
{

int var = 10;
int *ptr1 = &i;
int *ptr2 = (int *)2000;

printf("%d",ptr1*var);

return 0;
}
```

Output :

```
Compile Error
```

Comparing Pointers

- Relational operators can be used to compare addresses in pointers
- Comparing addresses in pointers is not the same as comparing contents pointed at by pointers:

```
if (ptr1 == ptr2)    // compares
                     // addresses
if (*ptr1 == *ptr2) // compares
                    // contents
```

Pointers as Function Parameters

- **Pass-by-Value**
- In C/C++, by default, arguments are passed into functions *by value.*That is, a clone copy of the argument is made and passed into the function. Changes to the clone copy inside the function has no effect to the original argument in the caller. In other words, the called function has no access to the variables in the caller.

Pointers as Function Parameters

- **Pass-by-Reference with Pointer Arguments**
- In many situations, we may wish to modify the original copy directly (especially in passing huge object or array) to avoid the overhead of cloning. This can be done by passing a pointer of the object into the function, known as *pass-by-reference.*