Lecture 18

Pointers

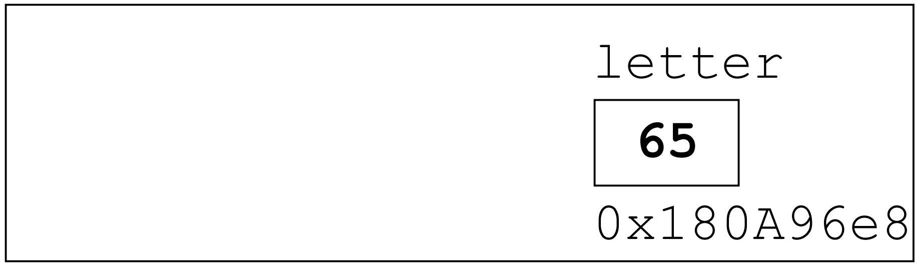
CSE115: Computing Concepts

What is a Pointer?

- So far, we have seen that a variable is used to store a value.
- Variables allow the programmer to directly manipulate the data in memory.
- A pointer variable, however, does not store a value but stores the **address of the memory** space which contain the value i.e. **it directly points to a specific memory address.**
- Why would we want to use pointers?
    - To call a function by reference so that the data passed to the function can be changed inside the function.
    - To create a dynamic data structure which can grow larger or smaller as necessary.

Variable Declaration

- A variable declaration such as,
  - char letter = ‘A’; causes the compiler to allocate a memory location for the variable *letter* and store in it the integer value 20.
  - This address of the memory location is available to our program during the run time.
  - The computer uses this address to access its content.



The name **`letter`** is associated with the address **`0x180A96e8`**

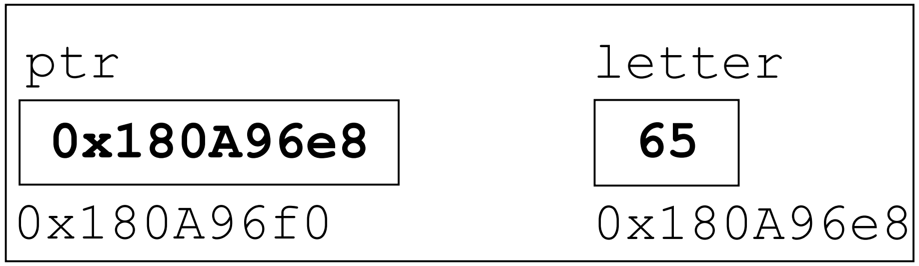
	address	content
	`0x00000000`	
	`0x00000001`	
	.	
	.	
	.	
`letter`	`0x180A96e8`	65
	`0x180A96e9`	
	`0x180A96f0`	
	.	
	.	

Pointers

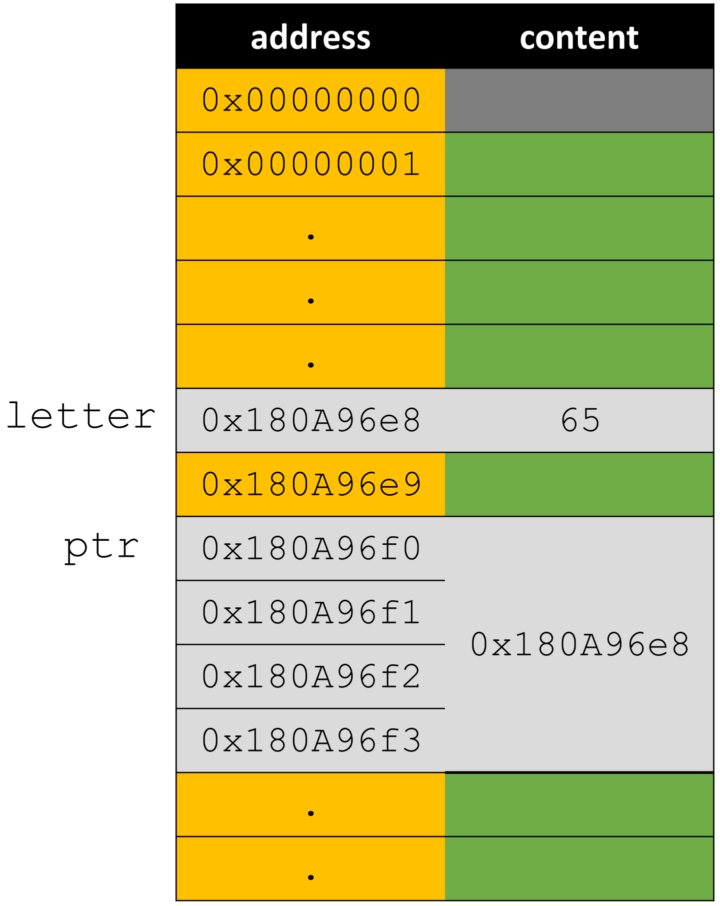
- A pointer is a variable that contains the address of another variable

char letter = ‘A’;

char *ptr = &letter;//ptr is a character pointer i.e. it can contain the address of a char type variable

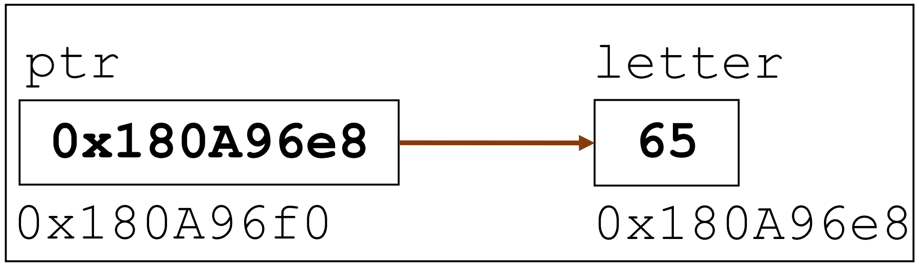


The variable **`ptr`** contains the address of **`letter`**



Pointers

- A pointer is a variable that contains the address of another variable
- We say that a pointer points/references another variable



The variable **`ptr`** contains the address of **`letter`**

	address	content
	0x00000000	▨
	0x00000001	🟩
	.	🟩
	.	🟩
	.	🟩
letter	0x180A96e8	65
	0x180A96e9	🟩
ptr	0x180A96f0	0x180A96e8
	0x180A96f1	
	0x180A96f2	
	0x180A96f3	
	.	🟩
	.	🟩

Pointer Declaration

- General Format:

  data_type *pointer_name;

- A pointer declaration such as,

  `int` **`*numberPtr;`**

  - declares `numberptr` as a variable that **points to an integer variable**. Its content is a **memory address.**

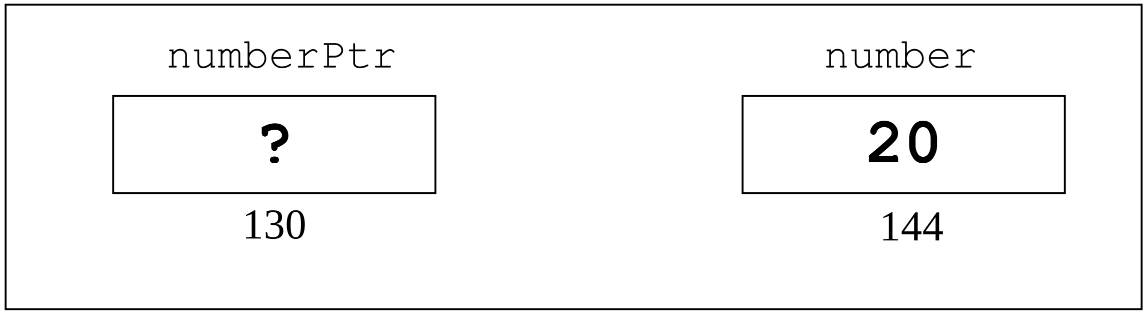
- The * indicates that the variable being declared is a pointer variable instead of a normal variable.

Pointer Declaration

- Consider the following declaration

```
int *numberPtr, number = 20;
```

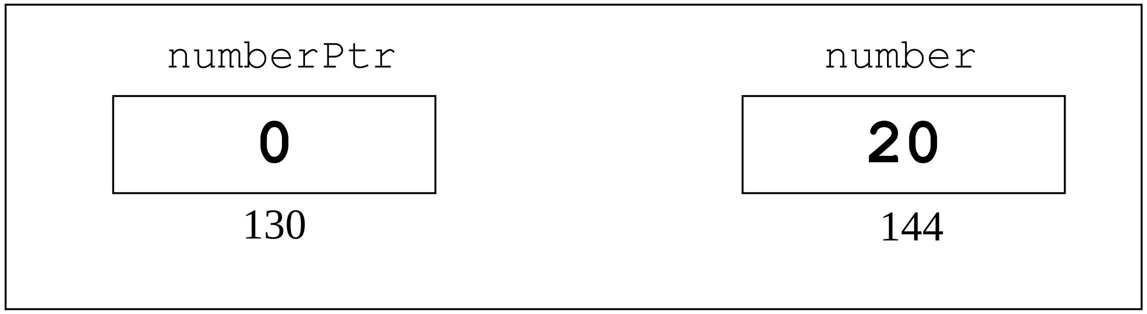
- In this case, two memory address have been reserved, associated with the names `numberPtr` and `number`.
- The value in variable `number` is of type integer, and the value in variable `numberPtr` is an address for another memory.



Pointer Initialization

- To prevent the pointer from pointing to a random memory address, it is advisable that the pointer is initialized to **NULL (the value 0) or an address** before being used.
- A pointer with the value NULL, points to nothing.
- Initializing a pointer to 0 is equivalent to initializing a pointer to NULL, but NULL is preferred.

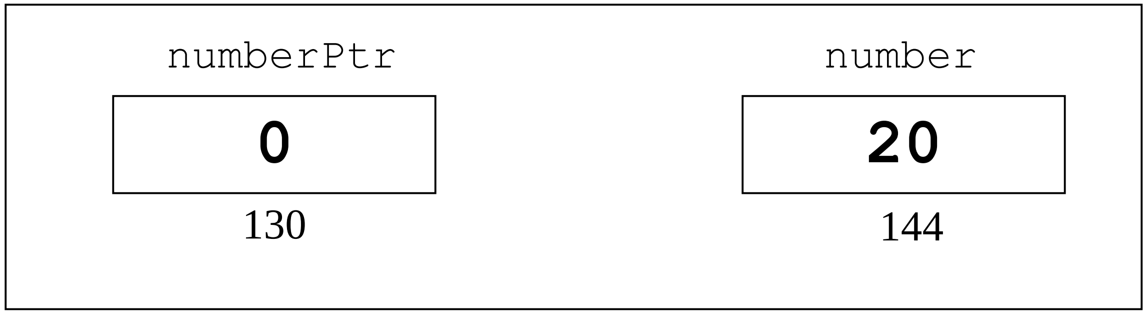
```
int *numberPtr = NULL;//equivalent to:
//int *numberPtr; numberPtr = NULL;
int number = 20;
```



Pointer Operator (& and *)

- When a pointer is created, it does not point to any valid memory address. Therefore, we need to assign a variable's address to it
  - using the **&** operator (**referencing operator/ address-of operator)**.
- Look at this example:

```
int *numberPtr, number = 20;
numberPtr = NULL;
```

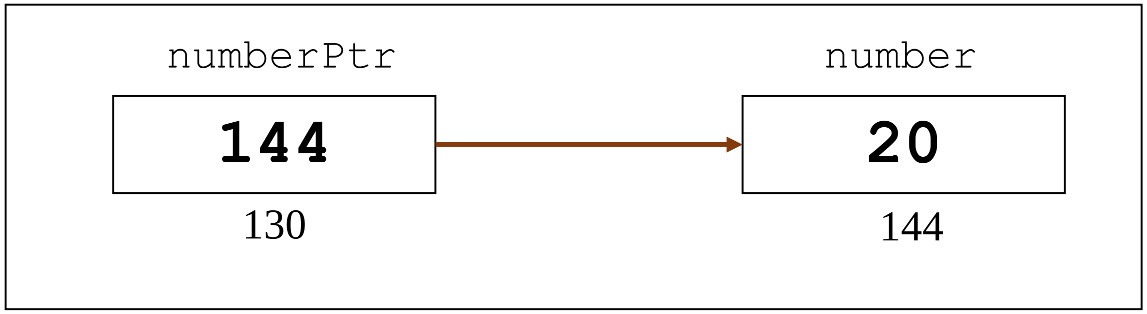


Pointer Operator (& and *)

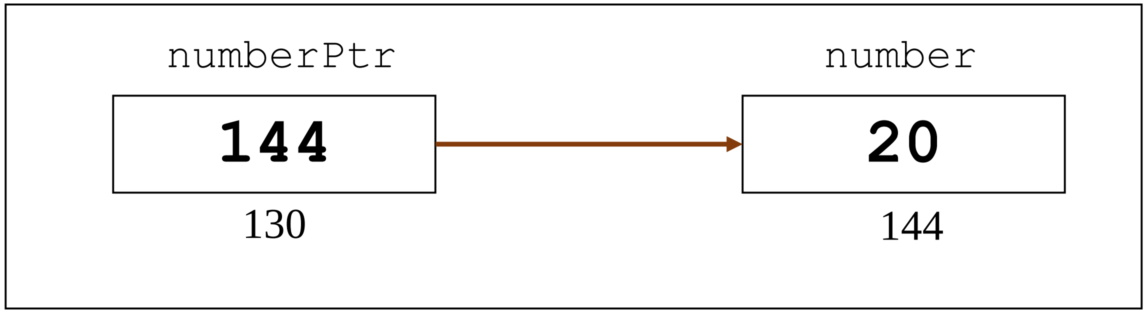
- When a pointer is created, it does not point to any valid memory address. Therefore, we need to assign a variable's address to it
  - using the **&** operator (**referencing operator/ address-of operator)**.
- Look at this example:

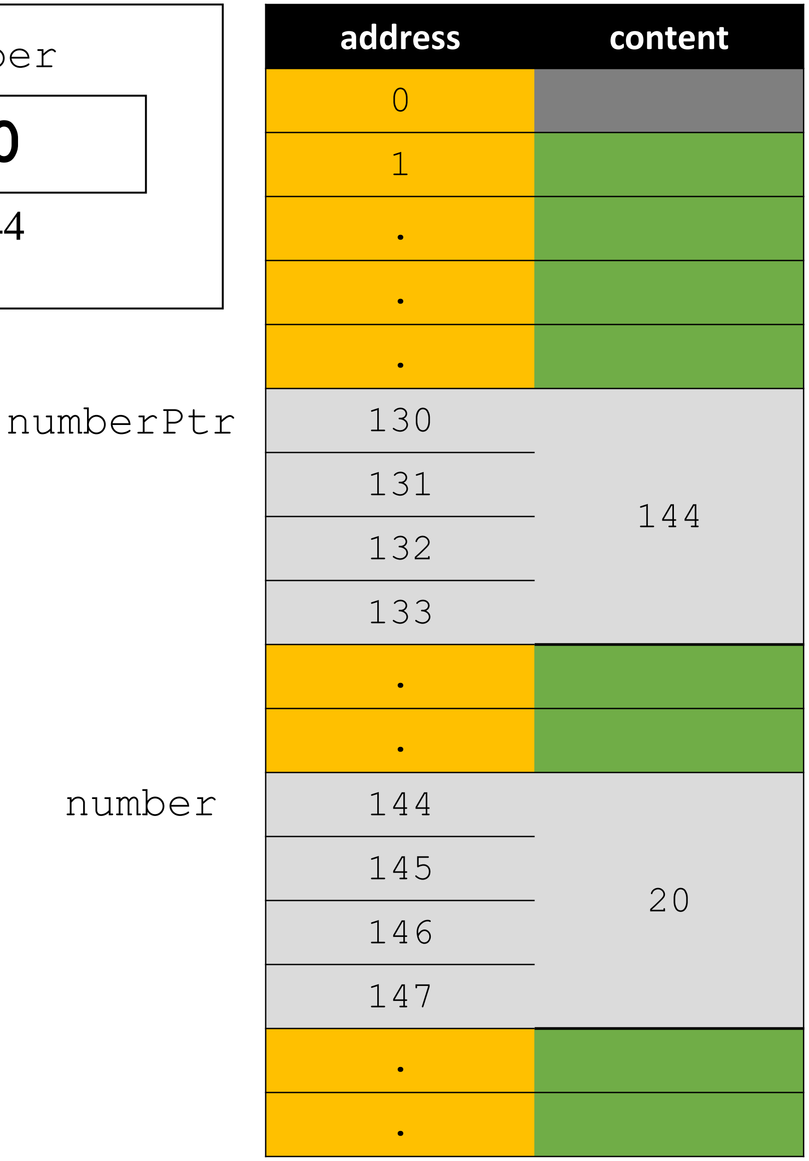
```
      int *numberPtr, number = 20;
      numberPtr = NULL;
      numberPtr = &number;
//address of number is assigned to numberPtr
```

- The statement **`numberPtr = &number`** assigns the address of the variable **`number`** to a pointer variable **`numberPtr`**. Variable **`numberPtr`** is then said as to "**point to**" variable **`number`**.



Pointer Operator (& and *)

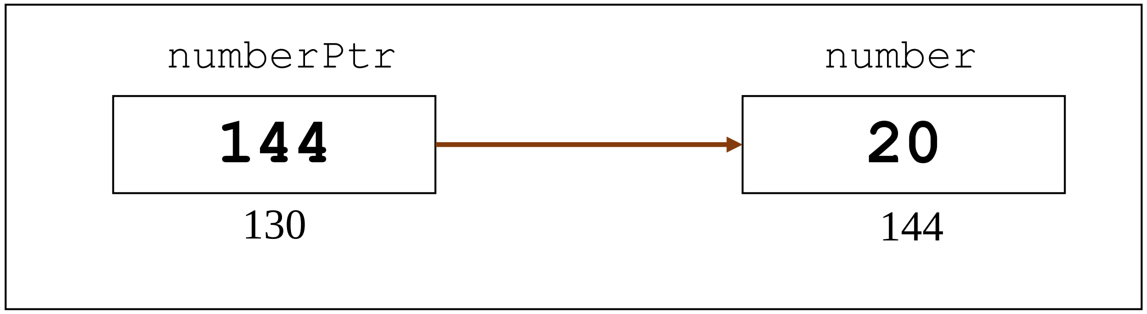




Pointer Operator (& and *)

- After a pointer is assigned a particular address, the value at the pointed address can be accessed/modified
  - using the * operator (**dereferencing operator/ value-at operator)**.
- Look at this example:

```
int *numberPtr, number = 20;
numberPtr = NULL;
numberPtr = &number;
```

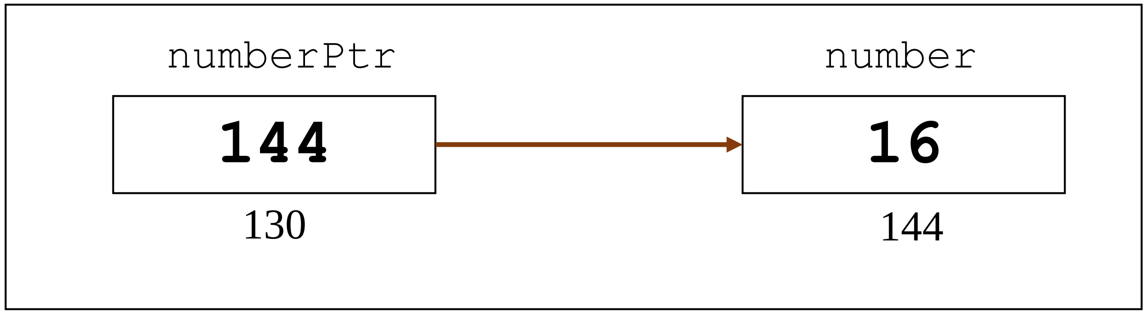


Pointer Operator (& and *)

- After a pointer is assigned a particular address, the value at the pointed address can be accessed/modified
  - using the * operator (**dereferencing operator/ value-at operator)**.
- Look at this example:

```
int *numberPtr, number = 20;
numberPtr = NULL;
numberPtr = &number;
*numberPtr = 16;//value at the address in numberPtr
     printf("number = %d", number);
```

- The statement *`numberPtr = 16` changes the content at the address 144 from 20 to 16.



Example: & and *

```
#include <stdio.h>
void main(void)
{
     int var = 10;
     int *ptrvar = &var;  //ptrvar = &var;

     printf("The address of the variable var is: %08x\n", &var);
     printf("The value of the pointer ptrvar is: %08x\n", ptrvar);
     printf("Both values are the same\n");

     printf("The value of the variable var is: %d\n", var);
     printf("The value of *ptrvar is: %d\n", *ptrvar);
     printf("Both values are the same\n");

     printf("The address of the value pointed by ptrvar is: %d\n",
  &(*ptrvar));//&(*ptrvar) == &var
     printf("The value inside the address of ptrvar is: %d\n",
  *&ptrvar);
     printf("Both values are the same\n");
}
```

Example: & and *

```
/*Sample Output */

The address of the variable var is: 1245052
The value of the pointer ptrvar is: 1245052
Both values are the same

The value of the variable var is: 10
The value of *ptrvar is: 10
Both values are the same

The address of the value pointed by ptrvar is: 1245052
The value inside the address of ptrvar is: 1245052
Both values are the same

Press any key to continue
```