CSE-4301
Object Oriented Programming
2022-2023

Week-7

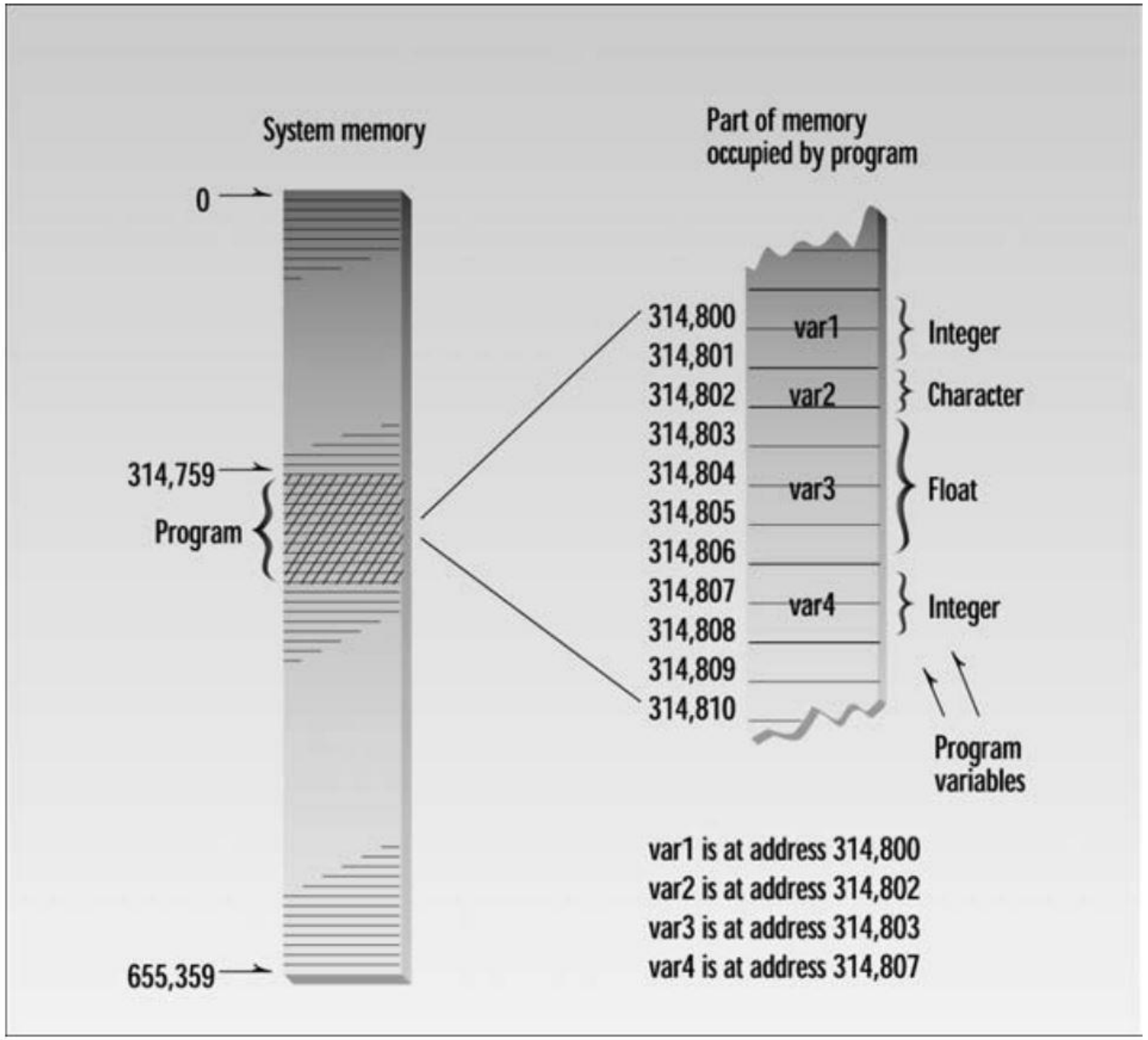
Pointers and Reference

-Faisal Hussain
Assistant Professor, Department Of Computer Science And Engineering, IUT

Contents

- **Addresses and Pointers**
- **The Address-of Operator** &
- **Pointers and Arrays**
- **Pointers and Functions**
- **Pointers and C-Type Strings**
- **Memory Management:** new **and** delete
- **Pointers to Objects**
- **A Linked List Example**
- **Pointers to Pointers**

Addresses and Pointers

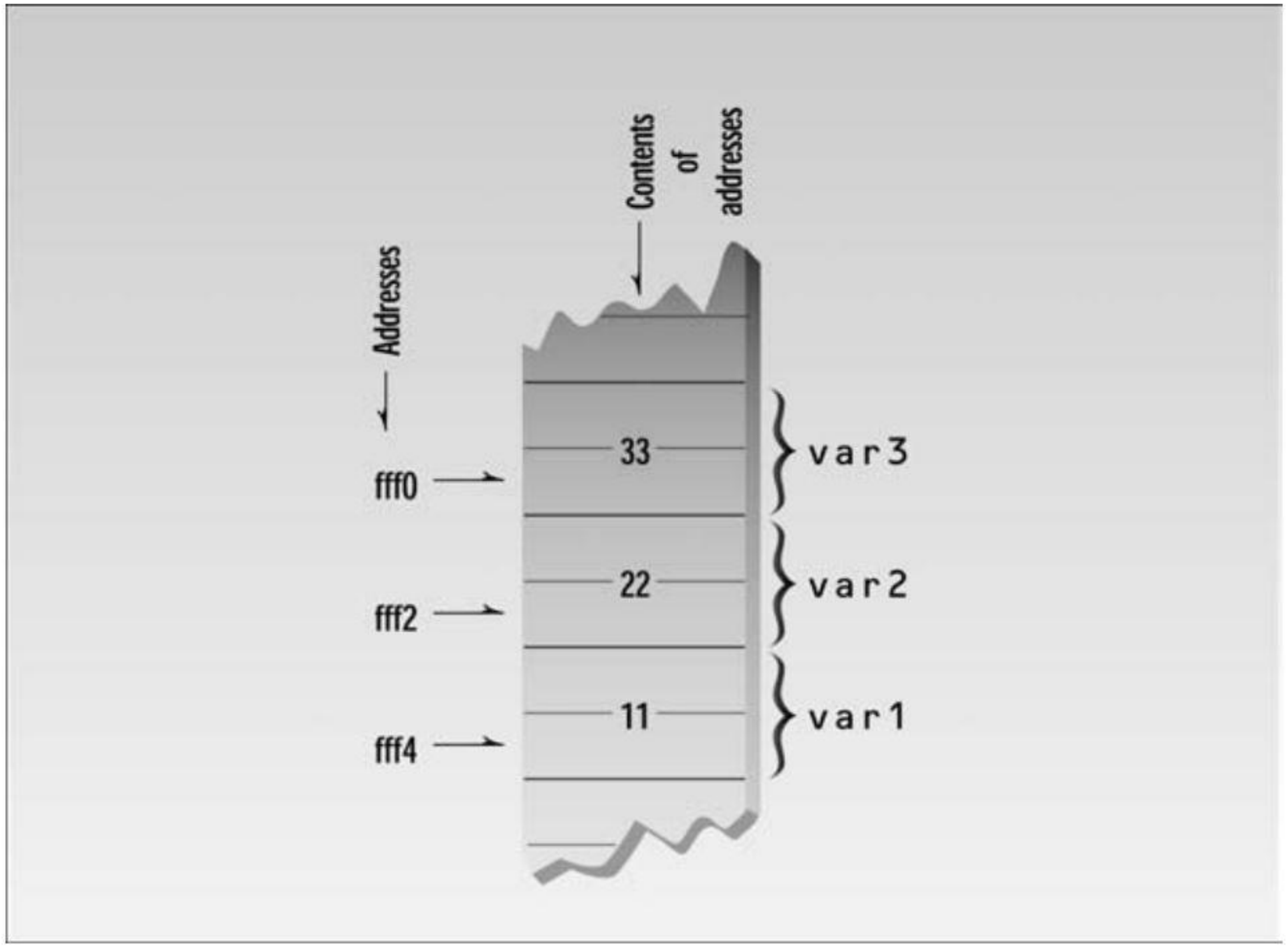


The Address-of Operator &

int var1;

cout<<&var1;

/* &var1 indicates the address of var1*/



Pointers

- A variable that holds an address value is called a *pointer variable*, or simply a *pointer*
- char* ptr1, * ptr2, * ptr3; // three variables of type char
  char *ptr1, *ptr2, *ptr3; // three variables of type char

- When an asterisk is used in front of a variable name, as it is in the *ptr expression, it is called the ***dereference operator*** (or sometimes the ***indirection operator***). It means *the value of the variable pointed to by*.

Pointers

- float flovar = 98.6;
  int* ptrint = &flovar;

//ERROR: can't assign float* to int*

- **Pointer to void**
  void* ptr;
  //ptr can point to any data type

- **The const Modifier and Pointers**

- const int* cptrInt;

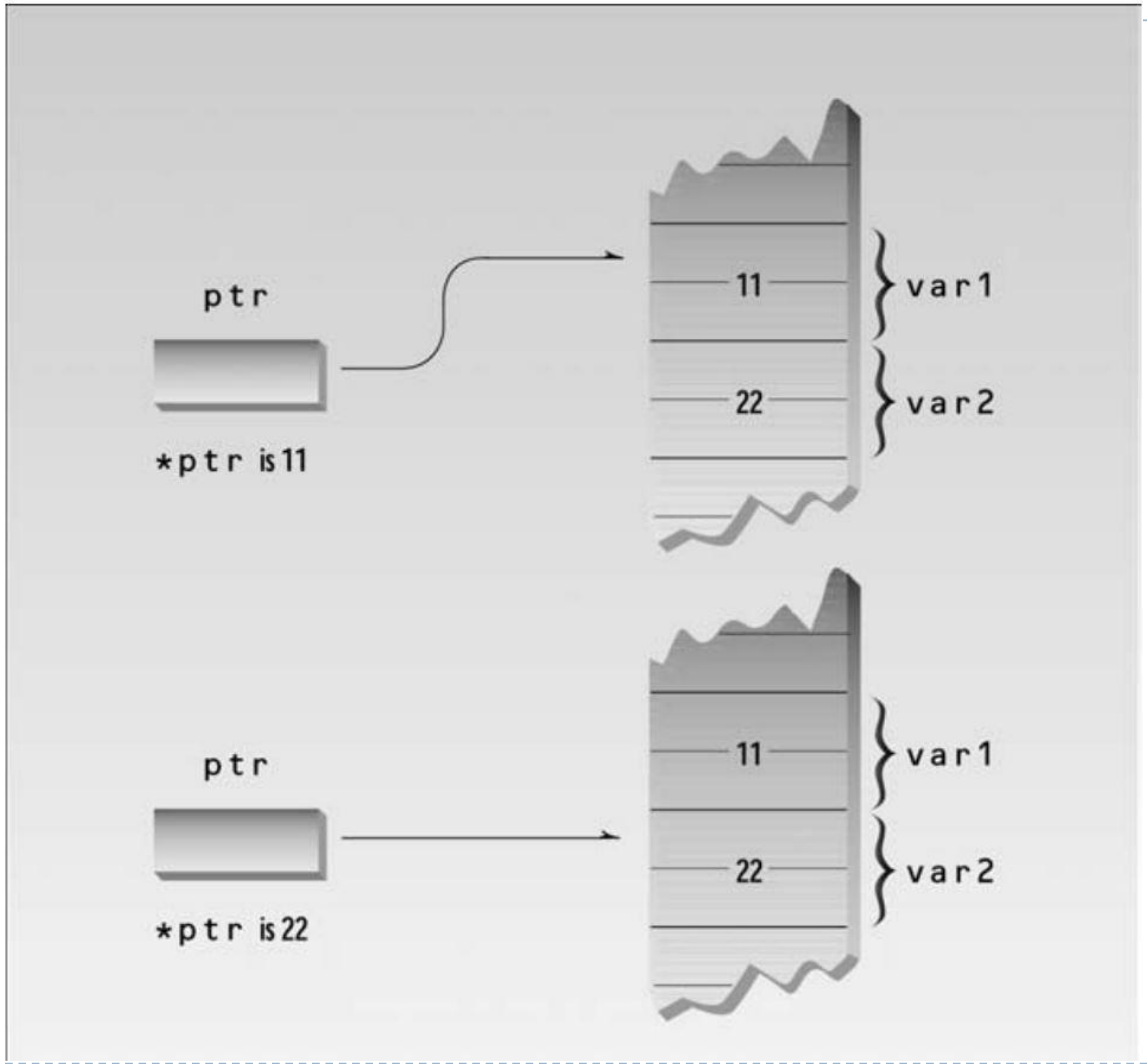
//cptrInt is a pointer to constant int

//using cptrInt you cannot assign any value.
//*cptrInt = 10;

int* const ptrcInt;

//ptrcInt is a constant pointer to int

//Once you initialize ptrcInt you cannot change



A Linked List Example

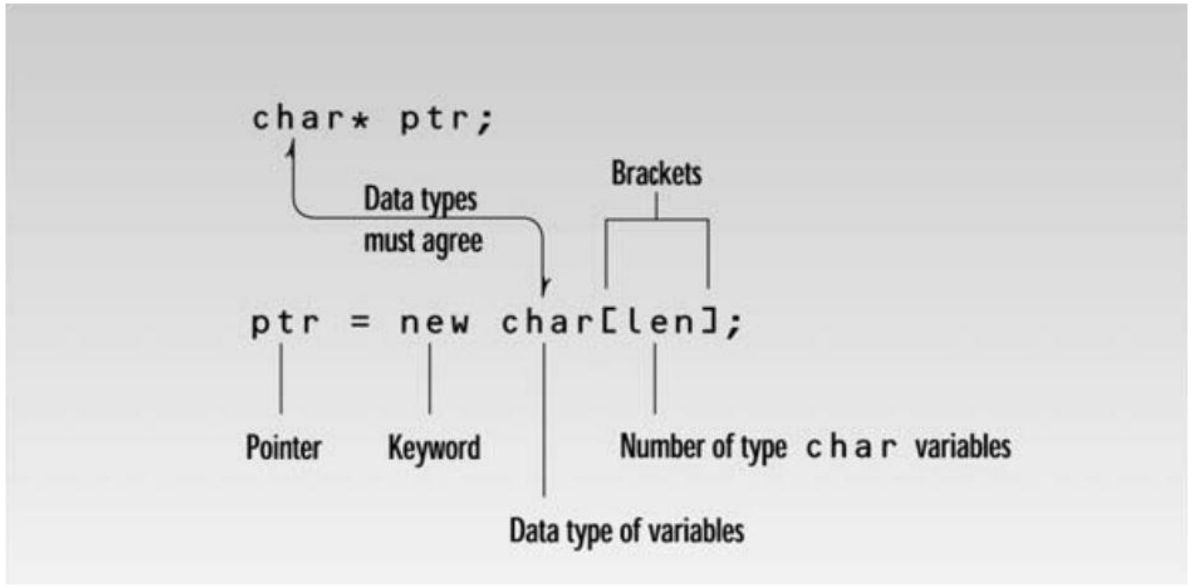
- Chain of pointer

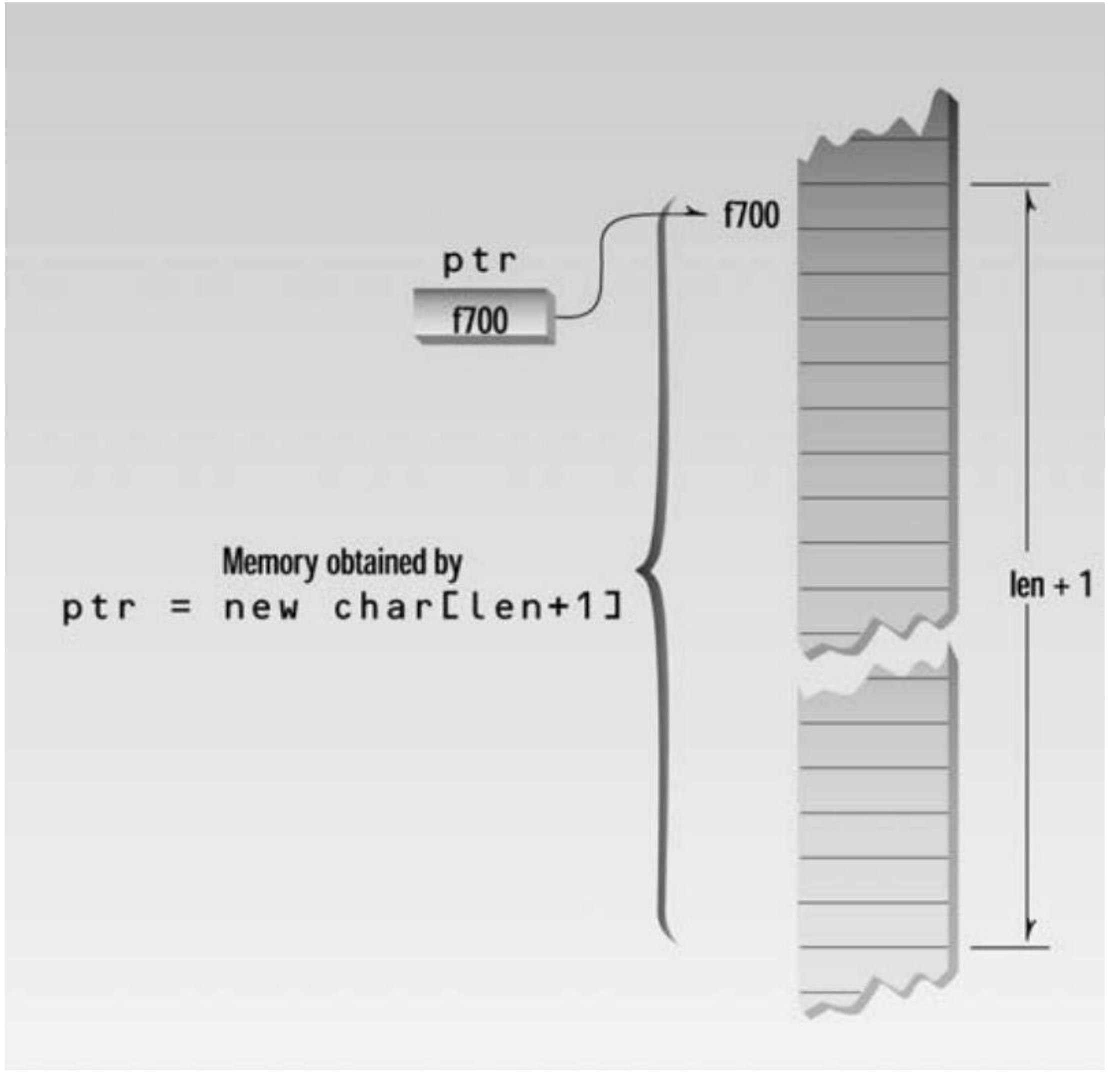
```
Struct link{
        int data;
        link * next;
}

Class linklist{
        private: link* first;
        ….
};
```

Self-Containing Classes

```
class sampleClass
{
        sampleClass* ptr;
};

class sampleClass
{
        sampleClass obj; /// Error
}
```

Memory Management: new and delete





The delete Operator

- Deleting the memory doesn’t delete the pointer that points to it (str in NEWINTRO), and doesn’t change the address value in the pointer.
- However, this address is no longer valid; the memory it points to may be changed to something entirely different. Be careful that you don’t use pointers to memory that has been deleted. .
- **Don’t forget the brackets when deleting arrays of objects**. Using them ensures that all the members of the array are deleted, and that the destructor is called for each one.

Pointers to Object/s

use the distance class code to understand

distptr.getdist(); // won’t work; distptr is not a variable
(*distptr).getdist(); // ok but inelegant

distptr->getdist(); // better approach

Creating a pointers do not trigger constructor.

use the pointer to objects code to understand

Pointers to Pointers

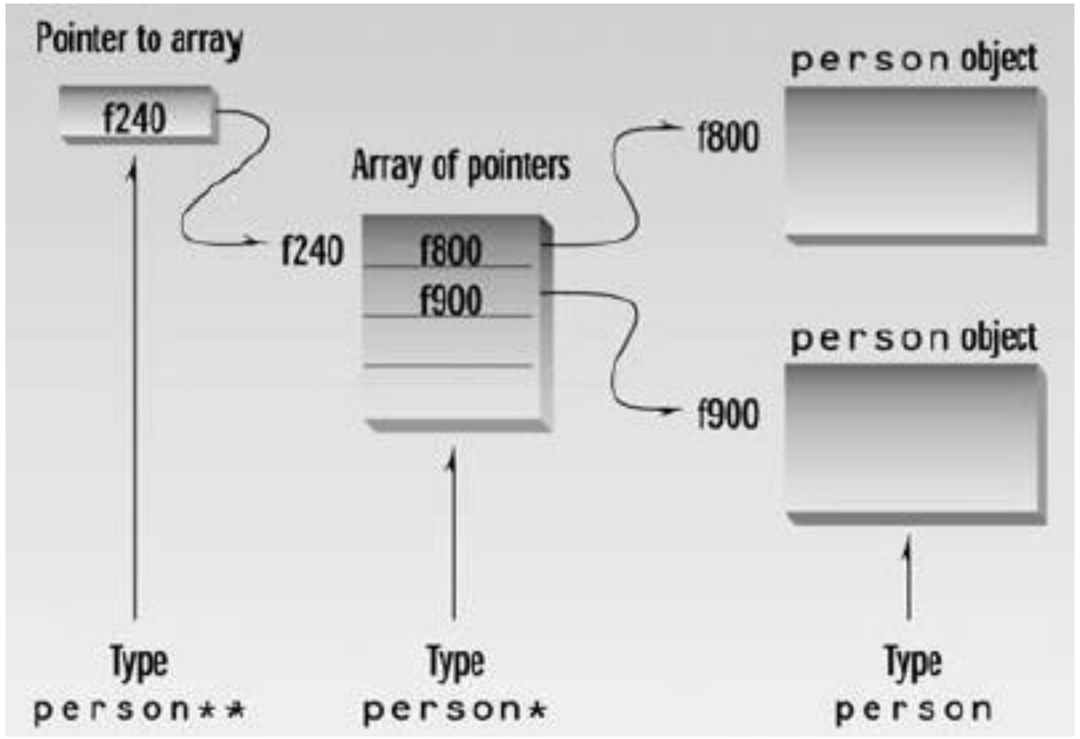
- ** use code to understand the use of Pointers to Pointers”

Person *p[10];

// An array of 10 pointers of Person class

Person **p2p = p;

//p2p pointer to pointer points to array of pointer



Reading Assignment

- A parsing Example
- Designing Horse Race

Page 481-489

Object Oriented Programming in C++ by Robert Lafore