CS601: Software Development for Scientific Computing

Autumn 2023

Week11: Intermediate C++, Structured Grids

References and Const

- We saw reference variables earlier
- Closely related to pointers:
  - Directly name another object of the *same* type.
  - A pointer is defined using the * (dereference operator) symbol. A reference is defined using the & (address of operator) symbol. Furthermore, unlike in pointer definitions, a reference must be defined/initialized with the object that it names *(cannot be changed later)*.

References

```
int n=10;
int &re=n; //re must be initialized
int* ptr; //ptr need not be initialized here
ptr=&n //ptr now initialized (now pointing to n)
int x=20;
ptr=&x; //ptr now pointing to x
re=x; //is illegal. Cannot change what re names.
printf("%p %p\n",&re, &n); // re and n are naming the
same box in memory. Hence, they have the same address.
```

`const`

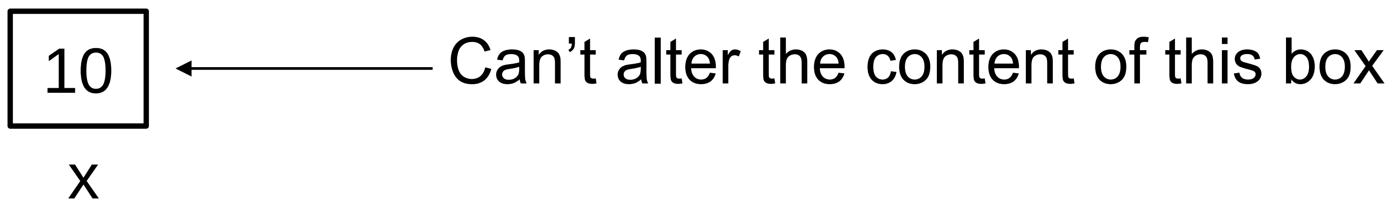
- A type qualifier
- The type is a constant (cannot be modified).
- `const` is the keyword
- Example:

```
const int x=10; //equivalent to: int const x=10;
//x is a constant integer. Hence, cannot be
//modified.
```

In what memory segment does x gets stored?

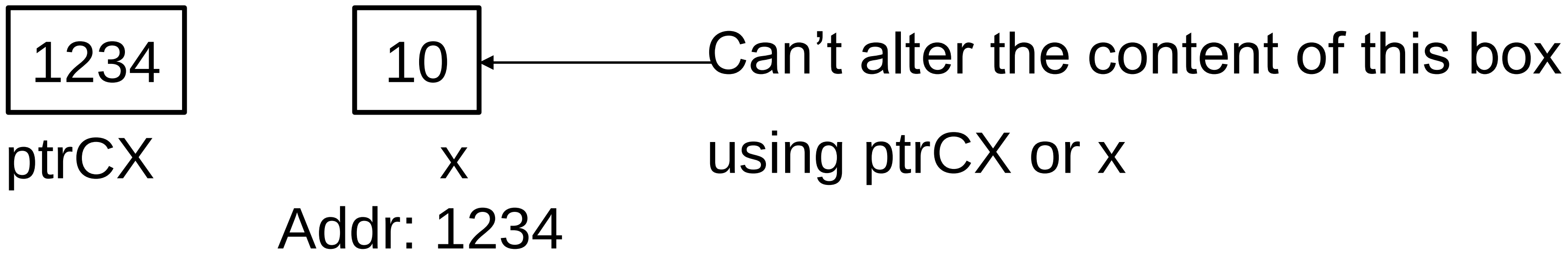
Const Properties

- Needs to be initialized at the time of definition
- Can’t modify after definition
- `const int x=10;`
  ```
  x=20; //compiler would throw an error
  ```
- `int const x=10;`
  ```
   x=10; //can’t even assign the same value
  ```
- `int const y; //uninitialized const variable y. Useless.`



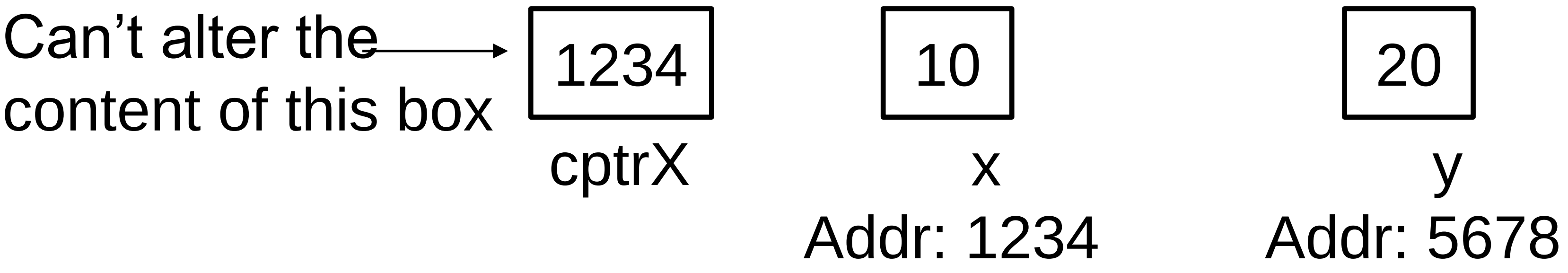
Const Example1 (error)

```
/*ptrCX is a pointer to a constant integer. So, can’t
modify what ptrCX points to.*/
const int* ptrCX; //or equivalently:
int const* ptrCX;

int const x=10;
ptrCX = &x;
*ptrCX = 20; //Error
```



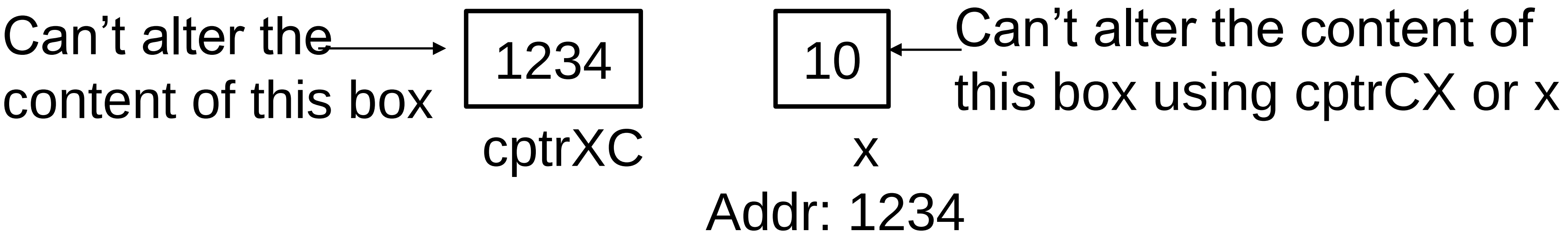
Const Example2 (error)

```
/*cptrX is a constant pointer to an integer. So, can’t
point to anything else after initialized.*/
int x=10, y=20;
int *const cptrX=&x;
cptrX = &y; //Error
```



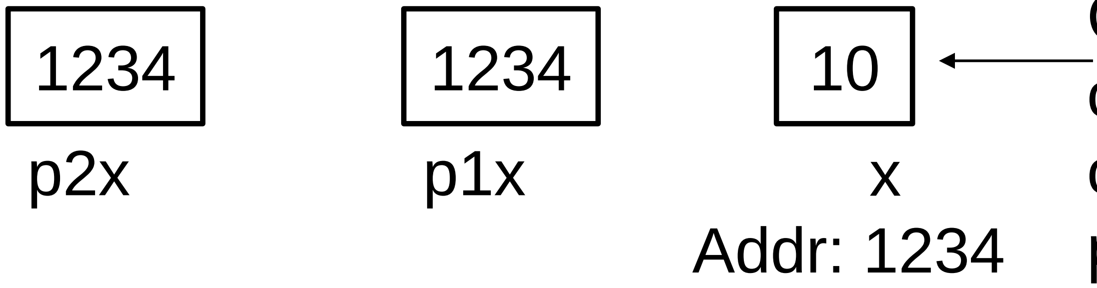
Const Example3 (error)

```
/*cptrXC is a constant pointer to a constant integer. So,
can’t point to anything else after initialized. Also,
can’t modify what cptrXC points to.*/

const int x=10, y=20;
const int *const cptrXC=&x;
int const *const cptrXC2=&x; //equivalent to prev. defn.
cptrXC = &y; //Error
*cptrXC = 40; //Error
```


Can’t alter the content of this box
1234
cptrXC
10
x
Addr: 1234
Can’t alter the content of this box using cptrCX or x

Const Example4 (warning)

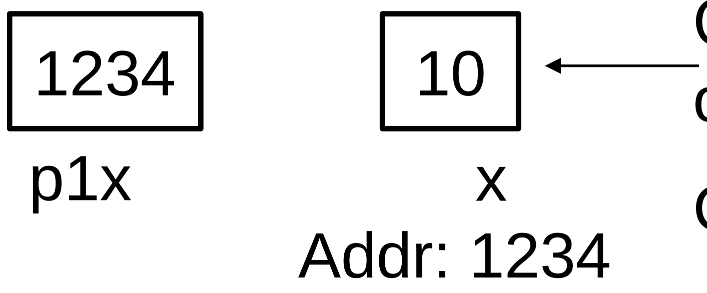
```
/*p2x is a pointer to an integer. So, we can use p2x to
alter the contents of the memory location that it points
to. However, the memory location contains read-only data -
cannot be altered. */

const int x=10;
const int *p1x=&x;
int *p2x=&x; //warning
*p2x = 20; //goes through. Might crash depending on memory
location accessed
```



Can’t alter the content of this box using p1x or x. Can alter using p2x.

Const Example5 (no warning, no error)

```
/*p1x is a pointer to a constant integer. So, we can’t use
p1x to alter the content of the memory location that it
points to. However, the memory location it points to can
be altered (through some other means e.g. using x)*/

int x=10;
const int *p1x=&x;
```

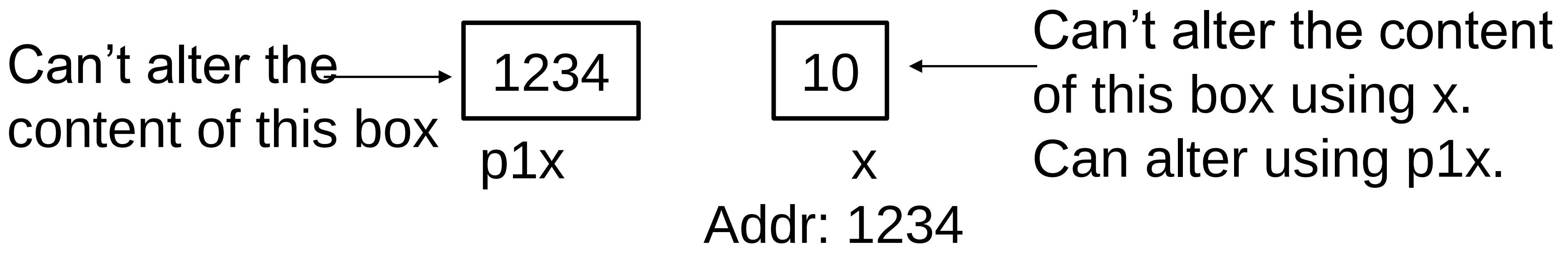


Can’t alter the content of this box using p1x.

Can alter using x.

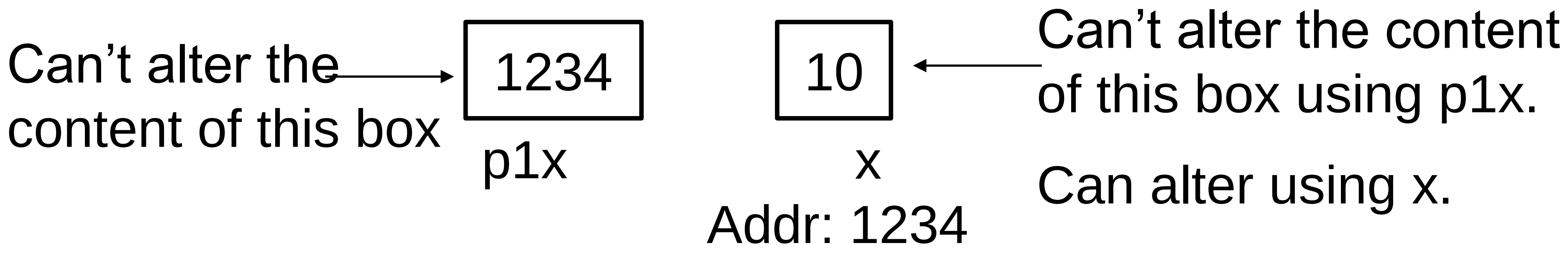
Const Example6 (warning)

```
/*p1x is a constant pointer to an integer. So, we can use
p1x to alter the contents of the memory location that it
points to (and we can't let p1x point to something else
other than x). However, the memory location contains read-
only data - cannot be altered. */

const int x=10;
int *const p1x=&x;//warning
*p1x = 20; //goes through. Might crash depending on memory
location accessed
```


Can't alter the content of this box
1234
p1x
10
x
Addr: 1234
Can't alter the content of this box using x. Can alter using p1x.

Const Example7 (no warning, no error)

```
/*p1x is a constant pointer to a constant integer. So, we
can’t use p1x to alter the content of the memory location
that it points to. However, the memory location it points
to can be altered (through some other means e.g. using
x)*/

int x=10;
const int *const p1x=&x;
```


Can’t alter the content of this box
1234
p1x
10
x
Addr: 1234
Can’t alter the content of this box using p1x.
Can alter using x.

Const and References - Summary

- Allow for compiler optimizations
  - pass-by-reference: allows for passing large objects to a function call
- Tell us immediately (by looking at the interface) that a parameter is read-only

Templating Functions and Classes

- Provide a recipe for generating multiple versions of the function/class based on the data type of the data on which the function/class operates upon

- Demo: