1.2 lots more C++

Thursday, September 1, 2022 9:46

TODAY: LOTS of C++

- while, for, functions
- fundamental C++ types
- pointers
- arrays
- dynamic arrays
- memory management

st_ ack diagrams

syntax for for loops:

```
for (<initialization>;
     <condition>;
     <update>)
  {
    <body of loop>
  }
```

checked before each iteration

update happens after each iteration

```
while (<condition>)
  {
    <body of loop>
  }
```

functions can be defined before main OR give a fn prototype before main and fn def later

prototype:

<return type> <fn name>

<return type> <fn name>
(<param type> <pname>);

fundamental C++ types

int, short, long } types of integers

float, double } types of floating pt numbers

strings

bool can be true or false

char character

void absence of type

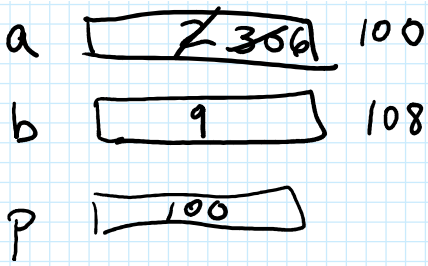
Pointers

Every variable is stored somewhere specific in the computer's memory.
its address

To obtain the address of a variable, do &<variable name>

For any type T, we can declare a pointer to a variable of that type by doing T* <pointer name>.

We can dereference a pointer to see what value is stored there by doing * <pointer name>.



Array

An array is a collection of elements of the same type.

To declare an array of type T, with N elements,

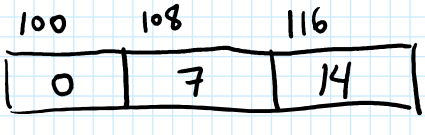
T <name>[N]; // This is static and its size is fixed.

ex: int arrayInts[101]; // a collection of 101 integers

Each element is referenced by its index, starting at zero.

arrayInts[0] ← the first element

The name of an array is equivalent to a pointer to the first element of the array.



Common mistake to AVOID:
indexing out-of-bounds.

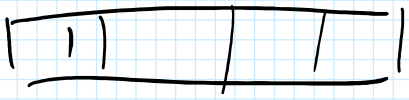
If we don't know how much memory we'll need to store the array, C++ lets us dynamically allocate it.

we'll need to store the array, C++ lets us dynamically allocate it.

Syntax:

T *<name> = new T[<size>];

This will return an address beginning the memory block of needed size.

dynamic_array 

For every new that appears in your program, there should be a corresponding delete to clean up the memory before the program ends.

The programmer must manage memory cleanup.

↓ only need for deleting arrays

delete [ ] <array name>;

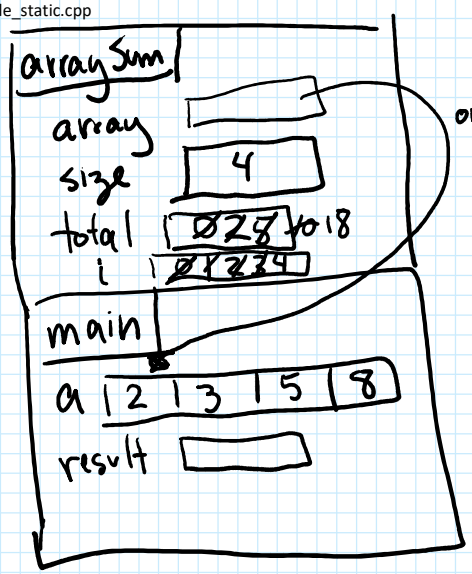
Stack diagram

A way to trace the execution of a program and its memory allocation.

- each function call adds a new frame to the stack
- static variables are drawn inside stack

- static variables are drawn inside stack frame where they're declared
- dynamic variables are stored in heap

stack_example_static.cpp



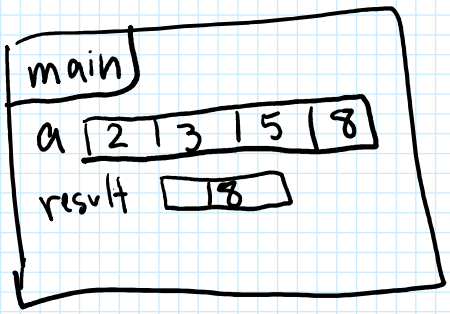
only a pointer! notice we did not copy over the whole array

STACK ↑

HEAP ↑ (Not used as this program is only static memory.)

later point:

← once we exit the function, its stack frame goes away



STACK ↑

HEAP ↑