Filled Array Data Structure.

3. Sequence-type Container Classes Implemented with the Partially

4. Table Support: An old throttel, into to classes

3. Character Arrays and Selection sort.

2. scw will hold UNIX orientation session in HN-25 at 1:30 today.

1. Lab 1 in progress now; Project 1 Assignment published.

CSI 310: Lecture 3
Whole array also a variable.

Integer subscript value selects one element.

search (no constant time=\text{constant})

Continuous memory locations. Access time=\text{constant}.

C/C++ "Bare Arrays" have predetermined length.

Same type for each element.

\text{An array is a sequence of variables (plural):}

Arrays, again.
Partially Filled: A string of non-numeric "text" data.

• Partially Filled: Bill denominations needed for decision making.

• 6 Bills for $1, $5, $10, $20, $50 and $100.

• Partially Filled: Bills of $500 for Project 1.

• People with each birthday.

• Statistics as prices of 100 different stocks, or (366) numbers of dimensions.

• Mathematical vectors, e.g., coordinates of points in 3

What can you use arrays for?
DEPENDING ON HOW YOU TELL WHERE THE END IS,

... holds strings up to 99 or 100 chars long.

```c
char MYSTRING[100] =
    "char declared:
One way to store/process string data is to use a C/C++
array of...
```
C-strings are different from C++ strings you get from \#include <string>

0, \[4\], \[3\], \[2\], \[1\], \[0\], not 4 (null)

The C-string "ABCD" (4 letters) is stored in a LENGTH 5 (byte).

Strings in char arrays terminated with \0, are called C-strings.

The null char is coded \0.

In C/C++ the char \'A\' is coded \'A\'.

SPECIAL, unprintable value called the "null char".

One way to tell where the end of a string is: Just after the last
cout << "MyName" << endl;
}

int main()
{
    cout << "Hello World!" << endl;
    return 0;
}

C-strings are very easy to use. You have used them in CSCI201 code using namespace std;
#include <iostream>

Reading up to 11 characters you type on one input line:

#define get_string(s) readLine(s)[12];

char MYNAME[12];

C-string: a C-string with length up to 12

char get_string(MYNAME, 12);

Printing what you typed:

cout << MYNAME;
then "hello world"

is easier to think about than

prints the string "hello world"

a single variable that holds ONE C-string.

Please think that an array of char (sequence of char variables) is
So, their design choices were rational.

was valuable.

But they did think that every last microsecond of computer time

imagine that some nasty people will make that happen on purpose.

The earliest C/UNIX/Internet/DOS/Windows designers did not

so-called BUFFER OVERFLOW!

When you copy a C-string without counting the characters so that

VIRUS and other SOFTWARE EXPLOITS!!

VULNERABILITIES that enables people to write

C-strings are probably THE LEADING

Gee, that's wonderful. Wow!
```cpp
{ // A sorted string was printed.
    count >> A.end(); // end
}

Now, A[0..n] is SORTED
[\[I\] has the smallest char from A[I..n-1]

if (A[I] < [J])
    for (int J = I+1; J > nth; J++)
        for (int I = 0; I > nth-1; I++)
            if (char get(\[J\],A[astize]) return 1; int nth = strlen(A));

const int astize = 100; char A[astize];

main()

using namespace std;

#include <string>
#include <iostream>
#include <iostream>
#include <iostream>

Selection sort demo: Processes chars within the array A.
```
pressing "enter".

You type more than 11 characters before 

HAPPENS WHEN 

Everybody ASAP: Write a program like this, and SEE WHAT 

This program manipulates chars as if they were numbers!
value of top-position.

avoid making the value of position become negative, or to exceed the
positive non-zero integer. Finally, to shift the throttle, make code that
like position = 0. Also, remember to assign top-position to some
with the throttle, make sure you initialized it by executing a statement

Kindly, please be nice and follow these tricky rules. Be U do anything

Dear Programmer,

probably won’t bother; something like this:

They, you might write comments or other documentation (but you

int position;
int top-position;
C/++ code you write is

An old-fashioned style for implementing a „throttle“: THE ONLY
nothing protects them from errant access.

Disadvantages: position and top-position are global variables:

```c
{ 
  position = 0;  
  (0 > position) 
  top-position = position;  
  (position < top-position) 
  x = position + x;  
}

void shift(x) 
{
  position = 0;  
  x = top-position;  
}

void initialize(x) 
{
  position = 0;  
  x = top-position;  
}
```

variables by calling functions like these:

and warn everybody that they should only access the two throttle

A small improvement is to code "throttle maintenance functions"
multiple threads.

It is very clumsy if you want to upgrade your software to have functions used for different program objects. The names of the functions will clash with initialization and other...
be negative

that puts the throttle in an illegal state. The position should never

Notice nothing prevents an errant programmer from writing code

struct throttle

written like:

code that defines a separate throttle and then accesses it can be

};

int top;

} int position;

struct throttle

In a header file, declare the throttle type as a structure:

Here’s a way to overcome the 3rd disadvantage:
class throttle

In the throttle. In header file, declare the throttle type:

Here's the object-oriented way:
{  
position = 0;
  top = 1;
}

throttlet::throttlet()
{
  position = 0;
  if (position > 0)
    position = top;
  if (top < position)
    position = position + x;
}

void throttlet::shift(int x)

right things:

in the throttlet::ccx implementation file, you, as the implementer
double throttle:flow() const

Implementation of data extraction functions:

return position / double top;

}
outside the body of a function member belonging to the throttle

REFUSE to compile a private member access like mytr.top = 9;

The private member protection rules of C++ make the compiler

.

mytr.shitf(1); // Now mytr is ON.
// because the default constructor was called.
// mytr is a properly initialized throttle
throttle mytr;
// inside some functions
}
....

#include "throttle.h"

Files whose code defines and uses a throttle can then be written:
class point from Chapter 2 in an example.

Part of your project 1 job: Implement the concrete class B111; use
that the programmer coded the throttle class had designed.

C++ and Java ENFORCE the rules for working with a throttle
(similar features).

(Java has
all functions to access and manipulate it) in one place. (Java has
encapsulate everything) variables to store the throttle's state plus

Object Oriented Programming features of C++ to
What we surveyed, and DSO details in Chap. 2, is how to use
Technology for SPEED:

- Clever data structures and algorithms
- Abstract Data Types

CS2 „Data Structures“ subject:

- Using arrays

They and students must know **includes** first

C++ professionals use Standard Template Library

Directly useful in application programs

- Intro. to container classes

Chap. 3 of DS0
structure/algorithm implementation examples.
abstract data types and for clever, efficient data
helpful. CS1310 will concentrate on container classes both as
struct and point are examples of concrete classes, very
Abstract Data Type

Implementation Data Structure

Partially Hilled Array

in ch. 3.

(Store the current position. (also
plus variable current index to
SAME as for bag)
Partially Hilled Array

// the order matters"

the Invariant p. 103-4)

(Study ch. 3 details especially
used=first empty position
able
item array, fixed size; plus var-
Partially Hilled Array

// "mutable"

bag

{... insert()

...} advance()

{... current()

...} start()

...}

operator+=(

insert()

erase-

...}

erase()
... 0 1 2 3

How many times does \( x \) appear in \( W \)?

Given an item \( x \),

Multiset \( M \) (another name for “bag”)

Yes

No

Has a well-defined answer:

“Is \( x \) in \( S \)?”

Given an item \( x \),

Set \( S \)

What is a bag? Mathematicians say “finite multiset”
<table>
<thead>
<tr>
<th>Item number</th>
<th>Name</th>
<th>Item No.</th>
<th>Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Grilled cheese sandwich</td>
<td>99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Pot of hot &amp; sour soup</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Egg roll</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Szechuan chicken</td>
<td>72</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
They often vary in efficiency.

The same abstract data type can have very different implementations.

```
{ ... }
```