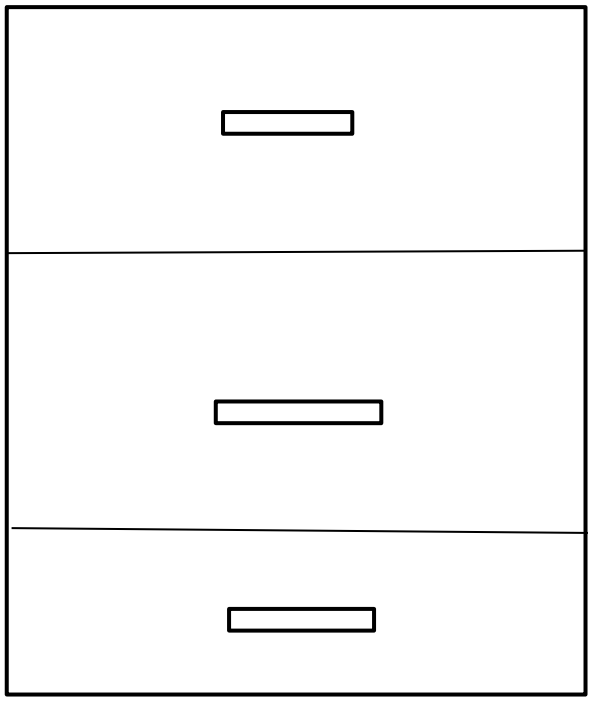
C Programming

Arrays

Situations will arise when you need to design a program that involves many variables of the same data type, e.g., lots of integers, lots of characters, etc.,

An array is a way to substitute having to declare multiple variables **of the same data type**.

Imagine an array is a filing cabinet where you can store multiple pieces of data in each drawer,



An array is what is known as a **Data Structure**

In C, you create an array as follows:

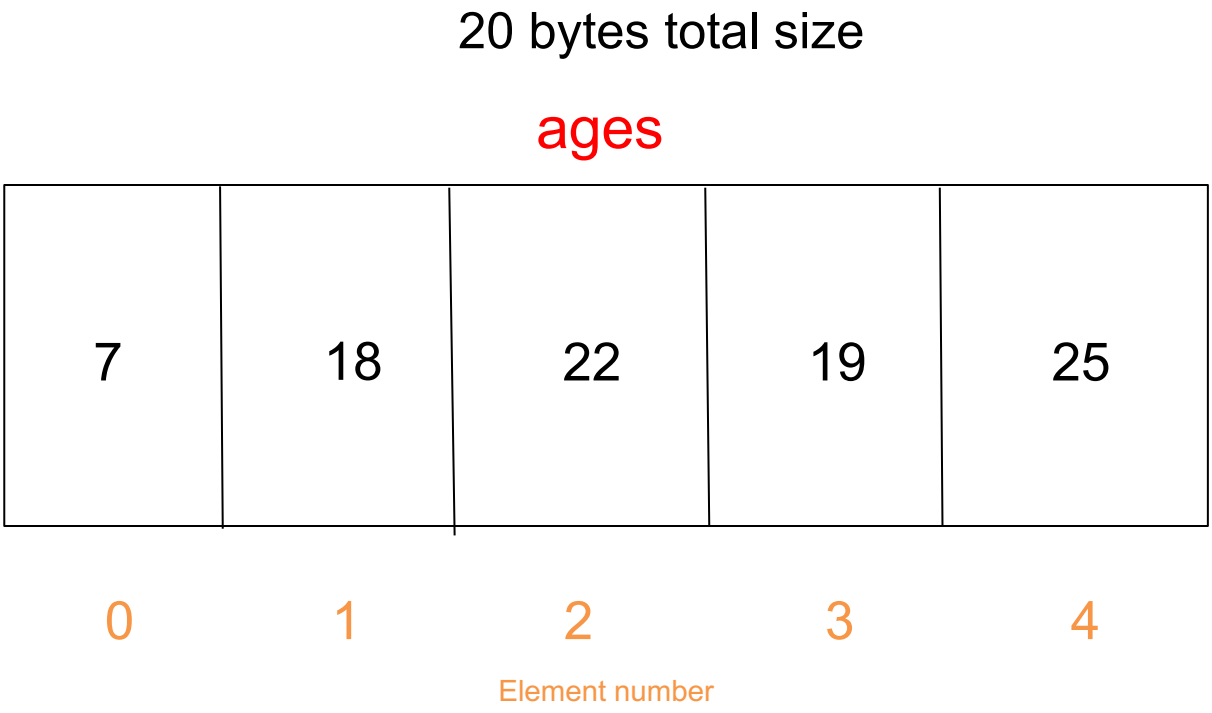
```
array_data_type  array_name [array_size];
```

e.g.,

```
int ages[5];
float number_set[10];
char name[12];
```

Let's have a look at the example below:

int ages[5];



ages[0] = 7;
ages[1] = 18;
ages[2] = 22;
ages[3] = 19;
ages[4] = 25;

```
/*
Program that uses an array and calculates the average age of a set of
people

*/
#include <stdio.h>

int main()
{
	int ages[5];
	int sum = 0;
	float average = 0;
	int i;
```

```
	printf("Enter the ages of 5 people\n");

	/*
	// This works but not realistic or efficient
	scanf("%d", &ages[0]);
	scanf("%d", &ages[1]);
	scanf("%d", &ages[2]);
	scanf("%d", &ages[3]);
	scanf("%d", &ages[4]);

	sum = ages[0] + ages[1] + ages[2] + ages[3] + ages[4];
	*/

	for(i = 0; i < 5; i++)
	{
		// enter an age
		scanf("%d", &ages[i]);

		// add the current value in sum to the age entered, i.e., keep a running sum total of ages
		sum = sum + ages[i];

	} // end for

	// cast the variable sum to be a float just for this line of code
	average = (float)sum / 5;

	printf("The average age is %.1f", average);

	return 0;

} // end main()
```

Repl 6.1: https://replit.com/@michaelTUDublin/61-1-D-array-p1#main.c

NOTE: There is one small limitation of an array in C. This is that an array can only store **data of the same type**

e.g.,
int numbers[10];
float averages[20];
char name[21]; // this is actually how we represent a string in C
short list[5];

Let's take a look at another good code example of an array. As follows:

```
/*
Program that uses an array and calculates the average age of a set of
people

*/
#include <stdio.h>

int main()
{
	int ages[5];
	int sum = 0;
	float average = 0;
	int i;
	int highest = 0;
	int lowest = 0;


	printf("Enter the ages of 5 people\n");

	// this loop will be used to enter an age into each element of
the array
	for(i = 0; i < 5; i++)
	{
		// enter an age
		scanf("%d", &ages[i]);

		// add the current value in sum to the age entered, i.e.,
keep a running sum total of ages
		sum = sum + ages[i];
```

```
	} // end for

	// Let's find the highest and lowest age entered
	// Assume the highest number in the array ages is inside element
0
	highest = ages[0];

	// Assume the lowest number in the array ages is inside element 0
	lowest = ages[0];

	/* this loop is used to go through each number in the array and
check if it is higher than the variable highest and lower than the
variable lower. If so, it replaces the values in highest and lowest
	*/
	for(i = 0; i < 5; i++)
	{
		// find the highest number in the array
		if(highest <  ages[i])
		{
			highest = ages[i];
		} // end if

		// find the lowest number in the array
		if(lowest > ages[i])
		{
			lowest = ages[i];
		} // end if

	} // end for

	// Display the highest and lowest number in the array
	printf("\n\nThe highest age is %d", highest);
	printf("\n\nThe lowest age is %d", lowest);

	// cast the variable sum to be a float just for this line of code
	average = (float)sum / 5;

	printf("\n\nThe average age is %.1f", average);
```

```
	return 0;

} // end main()
```

Repl 6.2: https://replit.com/@michaelTUDublin/62-1-D-array-p2#main.c

Symbolic Names

In C, a symbolic name is used to avoid hard-coding values in a program.

A symbolic name looks like this:

#define symbolic_name data_value

e.g.,

#define SIZE 5
#define LETTER 'A'

We normally use all uppercase characters for the names of symbolic names in order to visually differentiate them from regular variables in the program.

Let's substitute the above symbolic name SIZE into our previous program:

```
/*
Program that uses an array and calculates the average age of a set of
people

*/
#include <stdio.h>

// Symbolic name(s) are always listed below the last #include file
#define SIZE 10

int main()
{
	int ages[SIZE];
	int sum = 0;
	float average = 0;
	int i;
```

```
	int highest = 0;
	int lowest = 0;

	printf("Enter the ages of %d people\n", SIZE);

	// this loop will be used to enter an age into each element of
the array
	for(i = 0; i < SIZE; i++)
	{
		// enter an age
		scanf("%d", &ages[i]);

		// add the current value in sum to the age entered, i.e.,
keep a running sum total of ages
		sum = sum + ages[i];

	} // end for

	// Let's find the highest and lowest age entered
	// Assume the highest number in the array ages is inside element
	highest = ages[0];

	// Assume the lowest number in the array ages is inside element 0
	lowest = ages[0];

	/* this loop is used to go through each number in the array and
check if it is higher than the variable highest and lower than the
variable lower. If so, it replaces the values in highest and lowest
	*/
	for(i = 0; i < SIZE; i++)
	{
		// find the highest number in the array
		if(highest <  ages[i])
		{
			highest = ages[i];
		} // end if

		// find the lowest number in the array
		if(lowest > ages[i])
```

```
            {
                  lowest = ages[i];
            } // end if

      } // end for

      // Display the highest and lowest number in the array
      printf("\n\nThe highest age is %d", highest);
      printf("\n\nThe lowest age is %d", lowest);

      // cast the variable sum to be a float just for this line of code
      average = (float)sum / SIZE;

      printf("\n\nThe average age is %.1f", average);

      return 0;

} // end main()
```

Repl 6.3: https://replit.com/@michaelTUDublin/63-Symbolic-name#main.c