Computer Programming with

C++

Baby Steps

Presentation By AwesomeKen

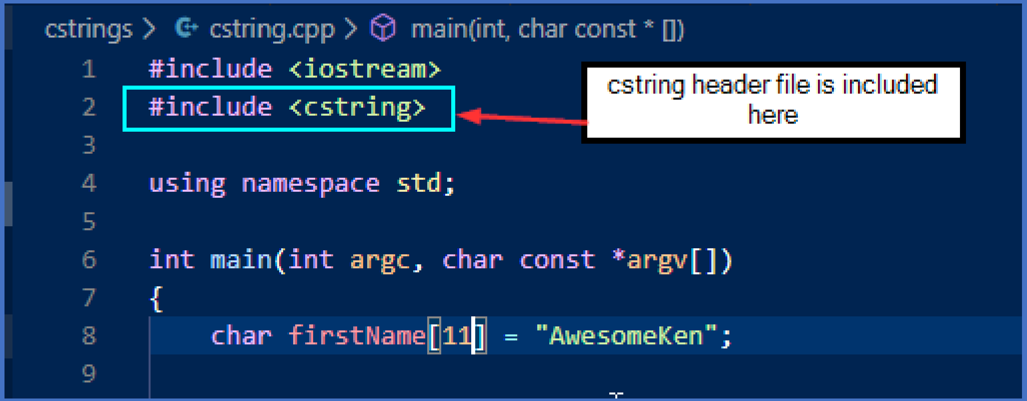
Outline

- Cstring header file
- C++ string
- Functions
- Errors

Cstring header file

- The creator of the C- language have been generous to have created some useful and handy functions to work with c-style strings.
- These functions are stored in a file and must be included to our project so as to have access to them.
- The name of the file is “cstring.h”.
- To include it to our project… this is how it is done

  #include <cstring>



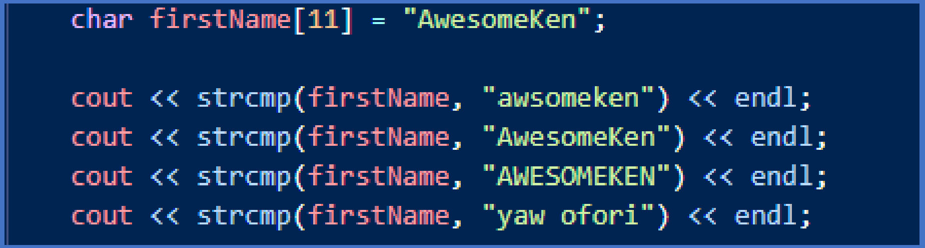
cstrings > cstring.cpp > main(int, char const * [])

```
#include <iostream>
#include <cstring>

using namespace std;

int main(int argc, char const *argv[])
{
    char firstName[11] = "AwesomeKen";

```

cstring header file is included here

Some Common functions from cstring.h

- Common operations performed on strings include :
- Comparing two string for equality. The function strcmp() is used.
- Strcmp(str1, str2) : returns a 0 if the two strings (str1 & str2) are equal and a negative number.



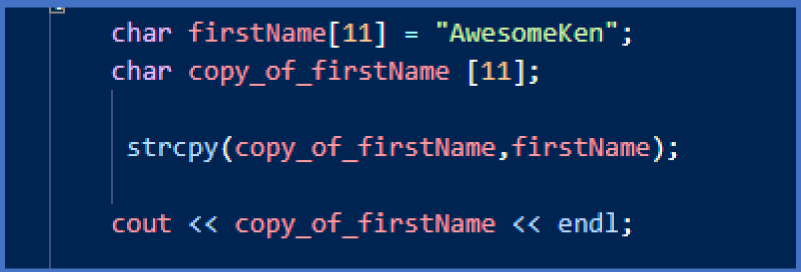
```
char firstName[11] = "AwesomeKen";

cout << strcmp(firstName, "awsomeken") << endl;
cout << strcmp(firstName, "AwesomeKen") << endl;
cout << strcmp(firstName, "AWESOMEKEN") << endl;
cout << strcmp(firstName, "yaw ofori") << endl;
```

```
Char firstName[11] = “AwesomeKen”;
```

Comparison	Results
Strcmp(firstName, “awsomeken”)	-32
Strcmp(firstName, “AwsomeKen”)	0
Strcmp(firstName, “AWSOMEKEN”)	32
Strcmp(firstName, “yaw ofori”)	-56

Some Common functions from cstring.h

- Common operations performed on strings include :
- Copying strings from one variable into another with strcpy().
- `strcpy(destination, source)` copy the content of one string to another
- Note `destination` must be of the same size or greater than the size of `source`



```
char firstName[11] = "AwesomeKen";
char copy_of_firstName [11];

 strcpy(copy_of_firstName,firstName);

cout << copy_of_firstName << endl;
```

- Others include strlen(), strstr(), strchar() etc.

C++ String

- C++ has a defined string type that comes with the comes with the compiler.
- This is an amazing string type as it relieves the programmer of having to deal with the trouble of the native c-strings.
- It can be included by using the include directive.
- #include <string>
- Declare a variable of type string.
- string var1; string var2;
- var1="The boy"; var2=" is going to school";
- To concatenate two strings simply: string var1_2 =var1 + var2;
- Equality check simply. var1 == var2;

Try Work

- Write a program to take the details of students in a university: the details include
- Name
- Age
- Student ID number
- Program of study
- Courses undertaking
- Level

Functions

- Lets look at a sample code of the strlen()
- The code is written in the main(). Now realize

this is just for firstName if we have other variables

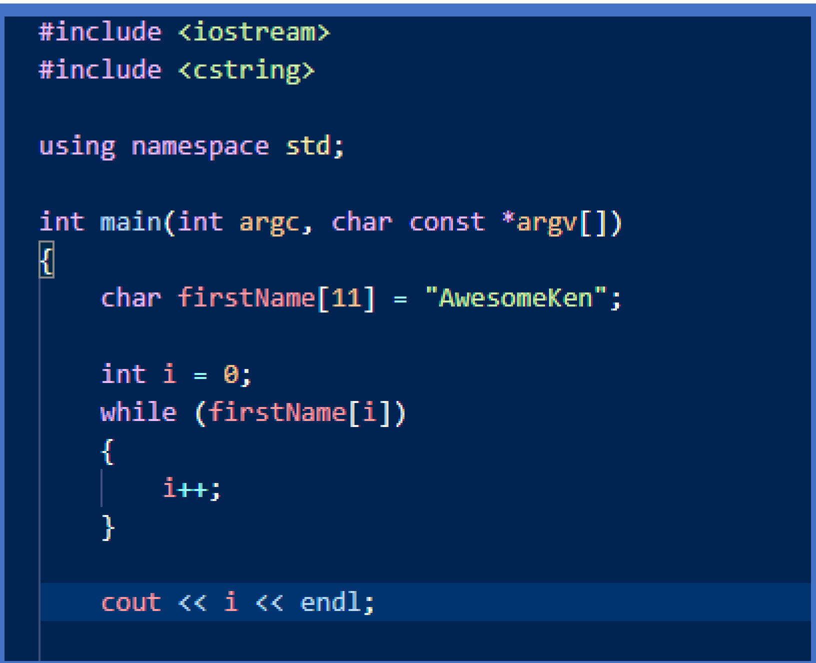
Each will have their while loop to count the length

Of the string.

- This is not very efficient. A very good and efficient code will be one in

which the code that counts the length is a package so that we don’t
have to write the count code each time we need it.

- A package of a sort is called FUNCTION.



```
#include <iostream>
#include <cstring>

using namespace std;

int main(int argc, char const *argv[])
{
    char firstName[11] = "AwesomeKen";

    int i = 0;
    while (firstName[i])
    {
        i++;
    }

    cout << i << endl;
```

Functions

Functions are created for specific purpose and after that is done a result of the purpose may be expected. Such a result is called the return value.

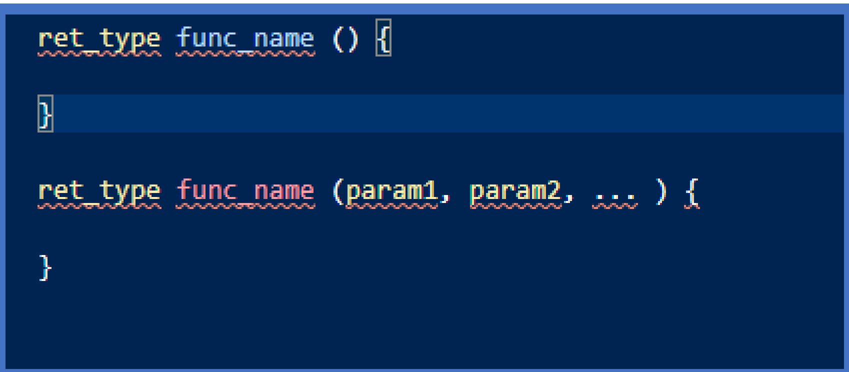
Ret_type – specifies the datatype of the value the function will return. Eg. int, double, char, char, string, float etc.*
Some functions do not return any value,in such cases the return type is specified as ***void***

Func_name – specifies the name of the function.

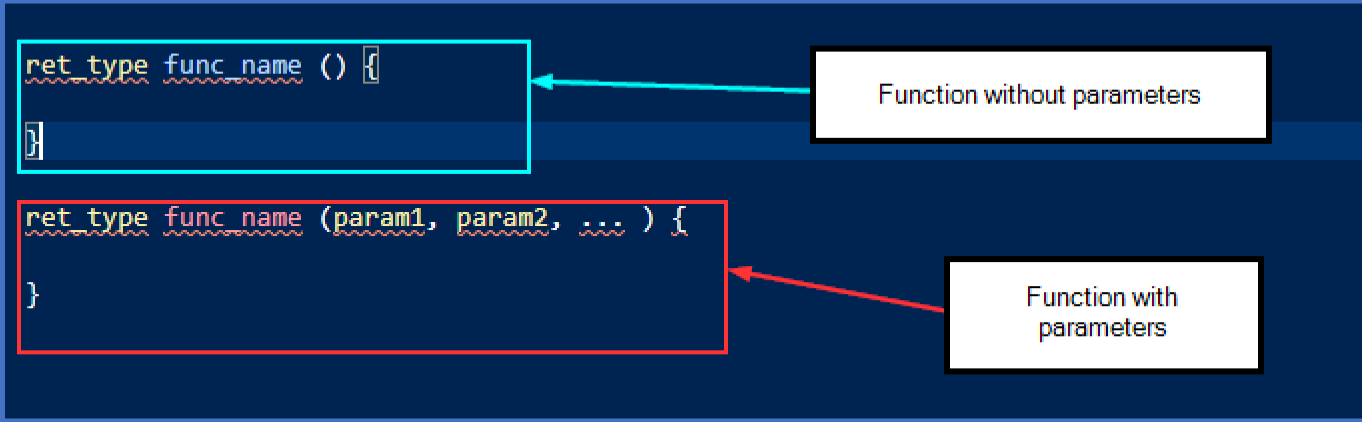
A function could have parameters – eg. A function that adds two numbers.

A function be without parameters – eg. A function that generates a random number.

Using a function is called calling a function.



```
ret_type func_name () {

}

ret_type func_name (param1, param2, ... ) {

}
```

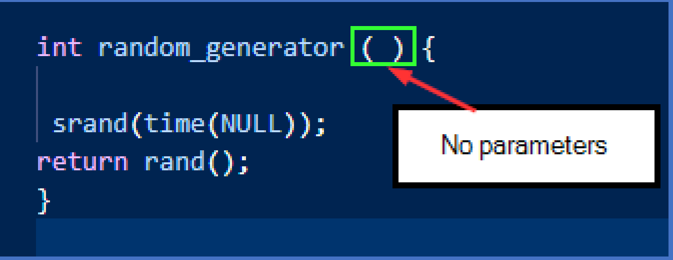


```
ret_type func_name () {

}
```

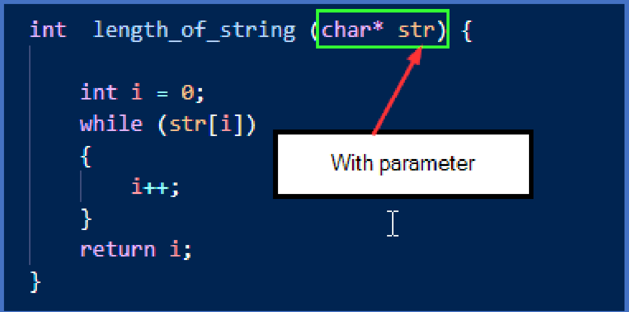
```
ret_type func_name (param1, param2, ... ) {

}
```

Functions

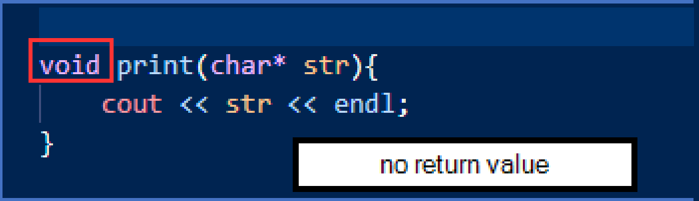
Different ways of creating functions.



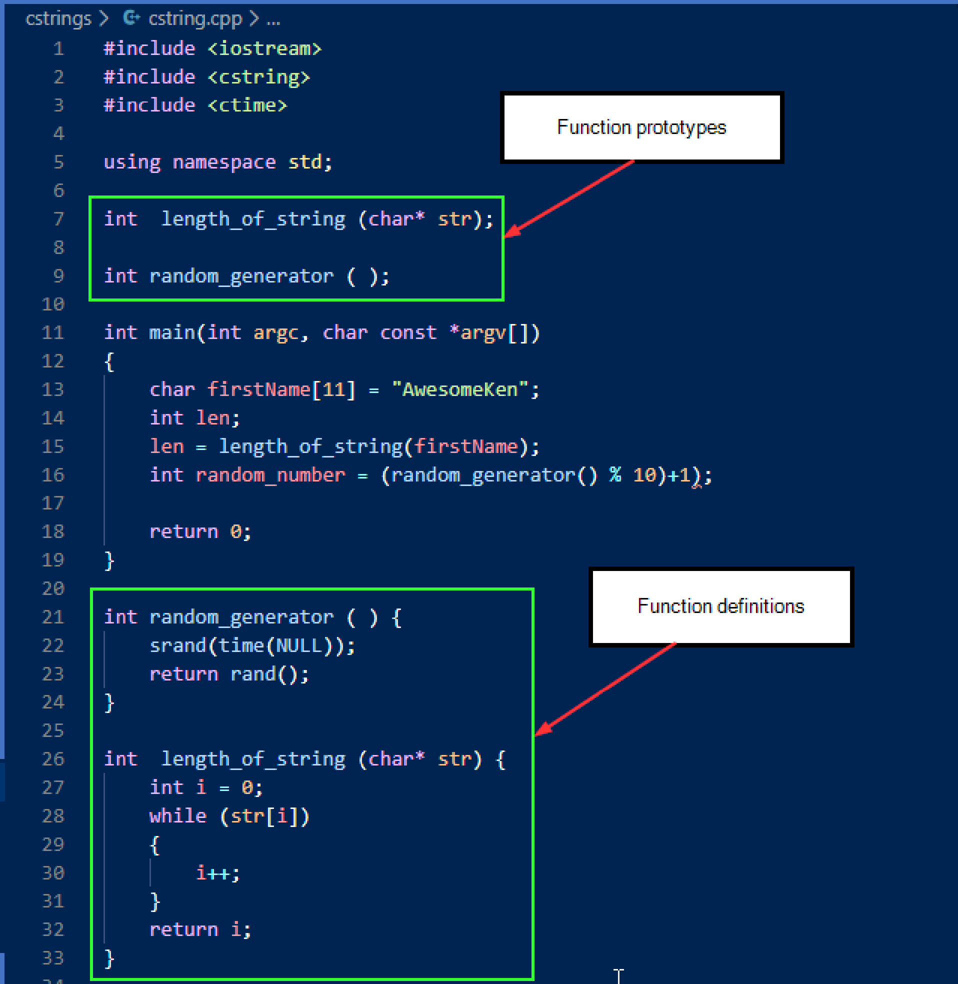
```
int random_generator ( ){

 srand(time(NULL));
return rand();
}
```



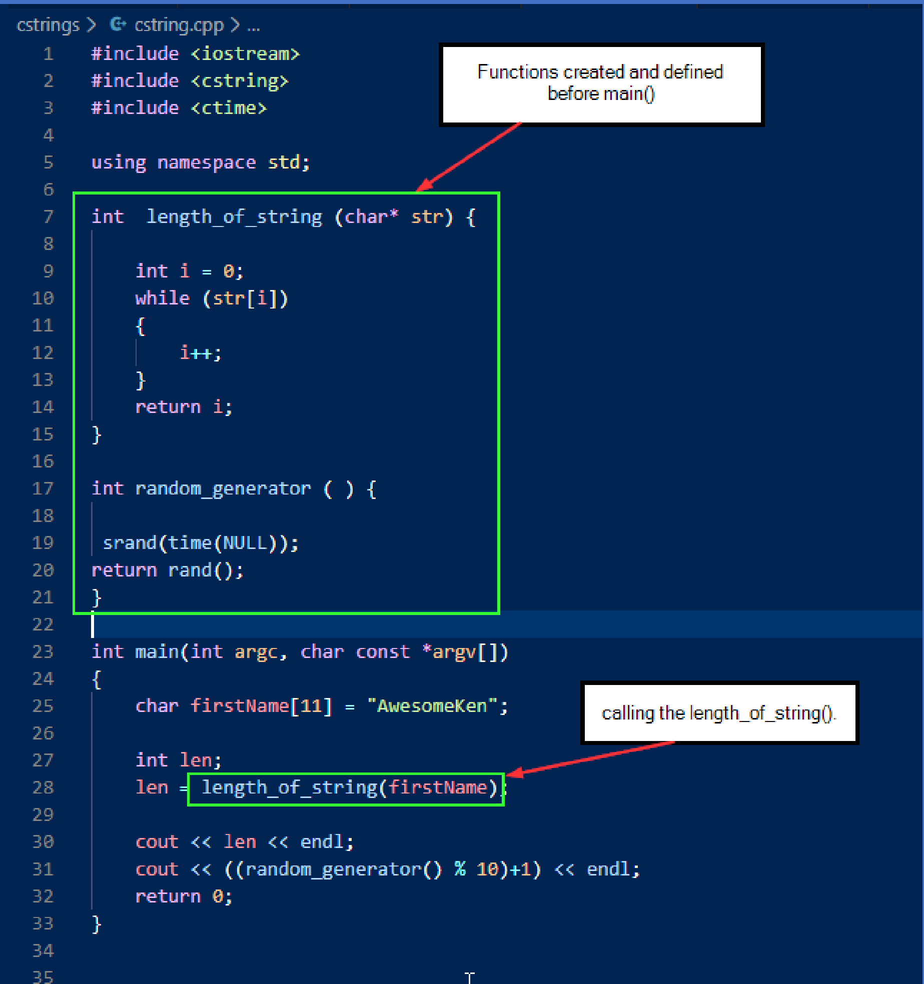
```
int  length_of_string (char* str) {

    int i = 0;
    while (str[i])
    {
        i++;
    }
    return i;
}
```



```
void print(char* str){
    cout << str << endl;
}
```



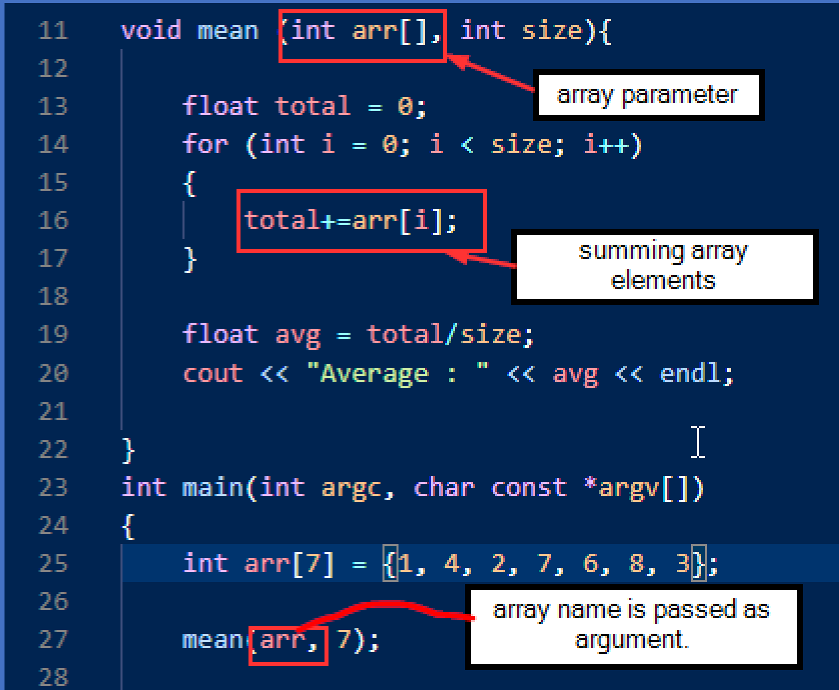
```
#include <iostream>
#include <cstring>
#include <ctime>

using namespace std;

int  length_of_string (char* str);

int random_generator ( );

int main(int argc, char const *argv[])
{
    char firstName[11] = "AwesomeKen";
    int len;
    len = length_of_string(firstName);
    int random_number = (random_generator() % 10)+1);

    return 0;
}

int random_generator ( ) {
    srand(time(NULL));
    return rand();
}

int  length_of_string (char* str) {
    int i = 0;
    while (str[i])
    {
        i++;
    }
    return i;
}
```


cstrings > cstring.cpp > ...
Functions created and defined before main()
calling the length_of_string().

```
#include <iostream>
#include <cstring>
#include <ctime>

using namespace std;

int  length_of_string (char* str) {

    int i = 0;
    while (str[i])
    {
        i++;
    }
    return i;
}

int random_generator ( ) {

 srand(time(NULL));
return rand();
}

int main(int argc, char const *argv[])
{
    char firstName[11] = "AwesomeKen";

    int len;
    len = length_of_string(firstName);

    cout << len << endl;
    cout << ((random_generator() % 10)+1) << endl;
    return 0;
}
```

Passing arrays to functions

- Arrays can be passed as arguments to a function.
- The function parameter has to be specified as to receive an array.
- Eg. `mean(int arr[], int size);` This function takes two arguments
    - An array of int's
    - The size of the array.
- When calling the function, the name of the array is passed as an argument to the function.



```
void mean (int arr[], int size){

    float total = 0;
    for (int i = 0; i < size; i++)
    {
        total+=arr[i];
    }

    float avg = total/size;
    cout << "Average : " << avg << endl;

}
int main(int argc, char const *argv[])
{
    int arr[7] = {1, 4, 2, 7, 6, 8, 3};

    mean(arr, 7);

```

Try Work

- Write a function that receives a string parameter and converts that string to uppercase.
- Write functions that check if
  - A number is odd.
  - If a number is even.
  - If a number is prime.
- Write a function that takes in the marks obtained and credit hours per course and calculates the CWA/ GPA of the student.

Errors

- `To err is human …`

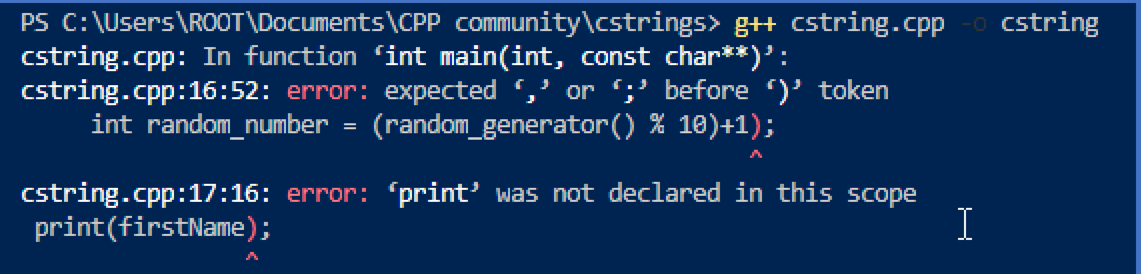
- As we go on this programming journey we are going to make a lot of errors as we write programs, knowing what type of error it is helps to find solution to it quickly.
- An error in programming is called a BUG and solving these errors is called DEBUGGING.
- Errors are categorized into
  - Syntax errors
  - Semantic errors
  - Runtime errors

Categories of Errors

Syntax Errors
- These errors are caught by the compiler.
- Errors like forgetting , or (or [or } or “ or ‘ or ;
- Undefined variables and functions.

Semantic Errors
- These errors are not caught by the compiler.
- Eg. If (k = 3) { return true; }
- This code will run perfectly since it is syntactically correct.
- However it is an error since it is not what the programmer intends.
- This sets k to 3 and since k is a positive number it will evaluate to true at all times.

Runtime Errors.
- The occur when the program is ran.
- They usually cause the program to crash.
- Eg. Trying to access out of bounds memory.
- Int ages[4]={3, 8, 2, 0};
- Ages[5] = 6;
- This will cause segmentation fault and cause the program to terminate abruptly.
- Divide by zero error.



```
PS C:\Users\ROOT\Documents\CPP community\cstrings> g++ cstring.cpp -o cstring
cstring.cpp: In function ‘int main(int, const char**)’:
cstring.cpp:16:52: error: expected ‘,’ or ‘;’ before ‘)’ token
     int random_number = (random_generator() % 10)+1);
                                                    ^
cstring.cpp:17:16: error: ‘print’ was not declared in this scope
 print(firstName);
                ^
```

Tips to Reducing Errors

- Plan before you program: it is not a song you are writing, its instructions and they require planning.
- Planning will help you to anticipate possible cases of errors to preempt them.
- Avoid programming when feeling sleepy
- Avoid programming when stressed or frustrated.
- Test your code frequently while programming: do not wait till the whole project is finished before you test your code. Test every function you write individually to make sure it is devoid of errors.
- Take a rest or walk if frustration sets in while programming.

Thank you