|  |  |  |
| --- | --- | --- |
| Writing a function; e.g., "cout << f", where f is a function name. | Attempt to write a function | 1st character of the function name. |
| Writing a struct name; e.g., "cout << P", where P is the name of a struct type. | Attempt to write a struct name | 1st character of the struct name. |
| Writing a struct variable; e.g., "cout << p", where p is a variable declared to be of a struct type. | Attempt to write a struct variable | 1st character of the struct variable. |
| Writing a void value (note: this can only happen if there is an attempt to write the return value from a void function); e.g., "cout << f()", where f is a void function. | Attempt to write void | 1st character of the function name. |
| Reading a function: e.g., "cin >> f", where f is a function name. | Attempt to read a function | 1st character of the function name. |
| Reading a struct name; e.g., "cin >> P", where P is the name of a struct type. | Attempt to read a struct name | 1st character of the struct name. |
| Reading a struct variable; e.g., "cin >> p", where p is a variable declared to be of a struct type. | Attempt to read a struct variable | 1st character of the struct variable. |
| Calling something other than a function; e.g., "x();", where x is not a function name. Note: In this case, you should *not* type-check the actual parameters. | Attempt to call a non-function | 1st character of the variable name. |
| Calling a function with the wrong number of arguments. Note: In this case, you should *not* type-check the actual parameters. | Function call with wrong number of args | 1st character of the function name. |
| Calling a function with an argument of the wrong type. Note: you should only check for this error if the number of arguments is correct. If there are several arguments with the wrong type, you must give an error message for each such argument. | Type of actual does not match type of formal | 1st character of the first identifier or literal in the actual parameter. |
| Returning from a non-void function with a plain return statement (i.e., one that does not return a value). | Missing return value | 0,0 |
| Returning a value from a void function. | Return with a value in a void function | 1st character of the first identifier or literal in the returned expression. |
| Returning a value of the wrong type from a non-void function. | Bad return value | 1st character of the first identifier or literal in the returned expression. |
| Applying an arithmetic operator (+, -, \*, /) to an operand with type other than int. Note: this includes the ++ and -- operators. | Arithmetic operator applied to non-numeric operand | 1st character of the first identifier or literal in an operand that is an expression of the wrong type. |
| Applying a relational operator (<, >, <=, >=) to an operand with type other than int. | Relational operator applied to non-numeric operand | 1st character of the first identifier or literal in an operand that is an expression of the wrong type. |
| Applying a logical operator (!, &&, ||) to an operand with type other than bool. | Logical operator applied to non-bool operand | 1st character of the first identifier or literal in an operand that is an expression of the wrong type. |
| Using a non-bool expression as the condition of an if. | Non-bool expression used as an if condition | 1st character of the first identifier or literal in the condition. |
| Using a non-bool expression as the condition of a while. | Non-bool expression used as a while condition | 1st character of the first identifier or literal in the condition. |
| Using a non-integer expression as the times clause of a repeat. | Non-integer expression used as a repeat clause | 1st character of the first identifier or literal in the condition. |
| Applying an equality operator (==, !=) to operands of two different types (e.g., "j == true", where j is of type int), or assigning a value of one type to a variable of another type (e.g., "j = true", where j is of type int). | Type mismatch | 1st character of the first identifier or literal in the left-hand operand. |
| Applying an equality operator (==, !=) to void function operands (e.g., "f() == g()", where f and g are functions whose return type is void). | Equality operator applied to void functions | 1st character of the first function name. |
| Comparing two functions for equality, e.g., "f == g" or "f != g", where f and g are function names. | Equality operator applied to functions | 1st character of the first function name. |
| Comparing two struct names for equality, e.g., "A == B" or "A != B", where A and B are the names of struct types. | Equality operator applied to struct names | 1st character of the first struct name. |
| Comparing two struct variables for equality, e.g., "a == b" or "a != b", where a and a are variables declared to be of struct types. | Equality operator applied to struct variables | 1st character of the first struct variable. |
| Assigning a function to a function; e.g., "f = g;", where f and g are function names. | Function assignment | 1st character of the first function name. |
| Assigning a struct name to a struct name; e.g., "A = B;", where A and B are the names of struct types. | Struct name assignment | 1st character of the first struct name. |
| Assigning a struct variable to a struct variable; e.g., "a = b;", where a and b are variables declared to be of struct types. | Struct variable assignment | 1st character of the first struct variable. |