Circumpage (Pop)	Sr No.	Parameters	С	C#
2. Developer			C is a Procedure Oriented	C# is an Object-Oriented
2. Developer Dennis Ritchie C is evolved from B, BCPL languages. It is a super sect of B. 4. Extension C is evolved from B, BCPL languages. It is a super sect of B. 4. Extension Approach C is evolved from B, BCPL languages. It is a super sect of B. C is evolved from B, BCPL languages. It is a super sect of B. C is evolved from B, BCPL languages. It is a super sect of B. C is evolved from B, BCPL languages. It is a super sect of B. C is evolved from approach. C is evolved to develop software. C is mostly used to develop software. C is mostly used to develop evolved in it is mostly used to develop web, mobile and destorp sophication. Beasic Building Block Finctions (C is function-driven) (C is fu			programming language (POP).	
perparaming programming Cris developer Dennis Ritchie Cris developed by Microsoft Co. Cell by Andres Heighergin 1200 Cris developed by Microsoft Co. Cell by Andres Heighergin 1200 Cris developed by Microsoft Co. Cell by Andres Heighergin 1200 Cris developed by Microsoft Co. Led by Andres Heighergin 1200 Cris developed by Microsoft Co. Led by Andres Heighergin 1200 Cris developed Cris developed Cris developed Cris developed Cris developed Cris mostly used to develope software. 8. Proximity to Hardware Cris more nearer to hardware than Cris Cris Mere Used Cris developed Cris Mere Used Cris Mortane. 9. Basic Building Block Function 10. Header file Supported in Cris Mere Cris Cris Mere Cris Cris Cris Cris Cris Cris Cris Cris				
2. Developer Dennis Ritchie C is evolved from B, SCPL languages. It is a super set of B. 4. Extension C itel have extension c C it seleved from B, SCPL languages. It is a super set of B. C it seleved from B, SCPL languages. It is a super set of B. C it seleved from B, SCPL languages. It is a super set of B. C it seleved from B, SCPL languages. It is a super set of B. C it seleved from B, SCPL languages. It is a super set of B. C it seleved from B, SCPL languages. It is a super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it seleved from B, SCPL languages. It is super set of B. C it is developed by used to develop software in the C if it is mostly used to develop web, mobile and destored in the C it is mostly used to develop web, mobile and destored in the C it is mostly used to develop web, mobile and destored in the A it is mostly used to develop web, mobile and destored in the A it is mostly used to develop web, mobile and destored in the A it is mostly used to develop web, mobile and destored in the C it is mostly used to develop web, mobile and destored in the C it is mostly used to develop web, mobile and destored in the C it is mostly used to develop web, mobile and destored in the C it is mostly used to develop web, mobile and destored in the C it is mostly us				
Sevolution C is evolved from B. BCPL languages. It is super set of B.	2.	Developer	Dennis Ritchie	
Languages Lang	2.	Severoper	Definis faccine	
4. Extension C. Ciles have extension C. Car files have extension C. Car files have extension C. Car uses bottom-up approach. 6. Execution Approach C. Used top-down approach. Car uses bottom-up approach. Car uses bottom	3.	Evolution	C is evolved from B, BCPL	•
4. Extension C Cfles have extension c 5. Execution Approach C used top down approach. C sees bottom wap paroach. C sees us so shower than C. T was so shower than C. C sees of the way show the sold seed to develop web, mobile and desktop application. C sees of the way show the sold seed of the problem into program and use to develop web, mobile and desktop application. C sees of the way show the sold seed of the problem into program and use to develop web, mobile and desktop application. C sees of the way show the sold seed of the problem into program and use to develop web, mobile and desktop application. C sees on thave close interaction with the hardware. (C is function-driven) (C is Delect threen) (C is post threen are 78 theywords and 31 (C is post threen) (C is post threen are 78 threen are				and Java too.
5. Execution Approach 6. Execution Speed 7. Where Used 7. Where Used 8. Proximity to Hardware 8. Proximity to Hardware 9. Basic Building Block 10. Execution is Sower and use to develop problem into angram and use to develop web, mobile and desktop application. 10. Header file 11. Keywords & Contextual keywords 12. Use of keyword as identifiers 12. Use of keyword as identifiers 13. Data Types 14. Size of Data Types 15. Integral Data Type 16. Data Security 17. Data Size integral by Data Is not secured. 18. Declaration Flexibility 18. Declaration Flexibility 19. Variable Name (identifier) length 20. Variable Name (identifier) length 21. Dealing with uninitialized 22. Automatic Coercions 23. Variable Name (identifier) length 24. Size Expression Outcome (return variables) 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays 28. Arrays 39. Research in C. Simported in C. Arrays are closured. 39. Arrays 30. Macros 30. Supported in C. Contormally supported in C. Arrays are of received in C. Arrays are for feeling of Central Central Research (Cop) for, while, do-while, forecast. 39. Supported in C. Arrays are for feeling in Central Research (Cop) for, while, do-while, forecast. 39. Supported in C. Arrays are for feeling in Central Research (Central Research Central Res	4	Extension	•	C# fligs have extension as
6. Execution speed 7. Where Used 8. Proximity to Hardware 9. Basic Building Block 10. Header file 11. Keywords & Contextual keywords 12. Use of keyword as identifiers 13. Data Types 14. Size of Data Types 15. Integral Data Type 16. Data Security 17. Data Visibility modes 16. Data Security 18. Declaration Flexibility 19. Variable Name (Identifier) length 19. Variable Name (Identifier) length 20. Dynamic Initialization of Variables 21. Use of integral by the initialized variables output. 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (foop) 25. Silent Fall-Through in switch statements (foop) 26. Supported in C. 27. Arrays 28. Arrays 29. Reported in C. 30. Memory Management 39. Politors 30. Support primitive data type 30. Data Visibility 30. Ces not provide any data visibility mode 30. Declaration Flexibility 30. Ces not provide any data visibility mode second as a declarations within a scope occur at the beginning of that scope 31. Data Types 32. Automatic Coercions 33. Memory Management 34. Automatic Coercions 35. Support find the same data type. 36. Automatic Coercions 36. Silent Fall-Through in switch statements (foop) 37. Automatic Coercions 38. Support din C. 39. Runtime Error handling (Exception Landling) 39. Memory Management 39. Supported in C. 39. Supported in C. 39. Supported in C. 39. Supported in C. 39. Runtime Error handling (Exception Landling) 39. Memory Management 39. Supported in C. 39. Runtime Error handling (Exception Landling) 39. Memory Management 39. Supported in C. 39. Suppor				
software, develop web, mobile and desktop application. 8. Proximity to Hardware than Cf. 9. Basic Building Block Functions (Cis function-driven) with the hardware. 10. Header file Supported in C Not supported by Cf. Contains 32 keywords (Cfs is object driven) 11. Keywords & Contextual keywords C Contains 32 keywords and 31 contextual keywords in Cf. 12. Use of keyword as identifiers Not Allowed C Permits use of keywords as identifier when they are prefixed with "G" characters in C." 13. Data Types Support primitive data type 14. Size of Data Types 15. Integral Data Type 16. Data Security 17. Data Wisibility modes 18. Declaration Flexibility 19. Data Wisibility modes 19. Variable Name (identifier) length scope occur at the beginning of that				
develop web, mobile and desktop application. 8. Proximity to Hardware than CF (is more nearer to hardware than CF) 9. Basic Building Block [Cis function driven) (Cis is Object as With the hardware with the hardware for the process of the process	7.	Where Used	C is mostly used to develop	It is mostly used to modal real life
8. Proximity to Hardware (Is more nearer to hardware than CW with the hardware.) 9. Basic Building Block Functions (Cis function-driven) (Cis fu			software.	, -
Basic Building Block				· · · · · · · · · · · · · · · · · · ·
Sasic Building Block	8.	Proximity to Hardware	C is more nearer to hardware	
Comparison Com		•	than C#	with the hardware.
11. Keywords & Contextual keywords 12. Use of keyword as identifiers 13. Use of keyword as identifiers 14. Size of Data Types 15. Use of Data Types 16. Data Types 17. Use of Late Types 18. Use of Late Types 18. Use of Late Types 19. Use Late Types 19. Use Late Types 19. Use Late Types 19. Use Late Late Late Late Late Late Late Lat	9.	Basic Building Block		•
11. Keywords & Contextual keywords 12. Use of keyword as identifiers 13. Data Types Support primitive data type 14. Size of Data Types Varied according to the machine ma	10	Hoodov file		
12. Use of keyword as identifiers 13. Data Types Support primitive data type 14. Size of Data Types Varied according to the machine of th				
12. Use of keyword as identifiers Not Allowed C # permits use of keywords as identifier when they are prefixed with "@" character. 13. Data Types Varied according to the machine machine 15. Integral Data Type There is only one integral type available in C 16. Data Security Data is not secured. Data is secured. There is only one integral type available in C Data Security Data is not secured. Data is secured. There is only one integral type available in C Data is secured. There is only one integral type available in C Data is secured. There is only one integral type available in C Data is secured. There is only one integral type available in C Data is secured. There is only one integral type available in C Data is secured. There is only one integral type available in C Data is secured. There is only one integral type available in C There is only one integral type available in C Data is secured. There is only one integral type available in C There is only one integral type available in C There is only one integral type available in C There is only one integral type available in C There is only one integral type available in C There is only one integral type available in C There is only one integral type available in C There is only one integral type available any data available in C There is only one integral type available any data available in C There is only one integral type available any data available in C There is only one integral type available any data available in C There is only one integral type available any data	11.	To a Contextual Reywords	5 contains 52 keywords	•
Support primitive data type All C# involves 2 integral types in it.	12.	Use of keyword as identifiers	Not Allowed	C# permits use of keywords as
13. Data Types Support primitive data type 14. Size of Data Types Varied according to the machine There is only one integral type available in C 15. Integral Data Type Data Sussey Data Is not secured. Data Security Data Visibility modes C does not provide any data visibility modes: Data Visibility modes C does not provide any data visibility modes: Protected, Internal, Internal Protected. Declaration Flexibility Inflexible, all variable declarations within a scope occur at the beginning of that scope Dynamic Initialization of Variables Des not support, a variable must be initialized using a constant expression. Des not support, a variable must be initialized variable remain undetected in C & thus gives unspredictable output. 22. Automatic Coercions Ves, with no warnings if loss could occur 23. Test Expression Outcome (return value) 124. Iteration statements (loop) 125. Silent Fall-Through in switch statements (loop) for, while, do-while, foreach. 226. Switch applicability Switch applicability Switch can be applied to integers & characters in C elements of the same data type. Arrays are of value type. Arrays are of value type. An array in C is not bound checked handling Memory Management Manual Marcos Supported in C Not supported in C Not supported in C# Supported in C Not supported in C Not supported in C Not supported in C Not supported in C#				*
14. Size of Data Types Varied according to the machine 15. Integral Data Type There is only one integral type available in C 16. Data Security Data Visibility modes C des not provide any data visibility modes C des not provide any data visibility modes C des not provide any data visibility modes 18. Declaration Flexibility Inflexible, all variable declarations within a scope occur at the beginning of that scope occur at the beginning of that scope 19. Variable Name (identifier) length 20. Dynamic Initialization of Variables Does not support, a variable must be initialized using a constant expression. 21. Dealing with uninitialized Variables Version Outcome (return value) 22. Automatic Coercions Ves., with no warnings if loss could occur 23. Test Expression Outcome (return value) 124. Iteration statements (loop) C has 3 iteration statements (loop) for, while, do-while. foreach integers a constant expression in the place of integers and loop for, while, do-while, foreach. 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays In C, arrays are of value type. Arrays are of reference types.	12	Data Types	Support primitive data to a	_
14. Size of Data Types Varied according to the machine Integral Data Type There is only one integral type available in C Data Security Data Integral Data Type Data Integral Data Type There is only one integral type available in C Data Security Data Integral Data Type Data Integral Type available in C Data Security Data Visibility modes C does not provide any data visibility modes: Public, Private, Protected, Internal, Internal Protected, In	13.	Data Types	Support primitive data type	
machine independent. There is only one integral type available in C Data Security Data is not secured. Cd involves 2 integral types in it. available in C Data Visibility modes C does not provide any data visibility modes: Public, Private, Protected, Internal, Internal Protected. Inflexible, all variable Protected, Internal Protected. Inflexible, all variable Protected, Internal Protected, Internal Protected. Inflexible, all variable Scope occur at the beginning of that scope occur at the beginning of that scope occur at the beginning of that scope. 20. Dynamic Initialization of Variables Dealing with uninitialized Uninitialized variable remain undetected in C & thus gives unpredictable output. At complete time. 21. Dealing with uninitialized Variables Variables 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (loop) C has 3 iteration statements (loop) C has 3 iteration statements (loop) for, while, do-while. 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability Arrays In C, arrays are the collection of elements of the same data type. Arrays or of value type. Arrays are of value type.	14.	Size of Data Types	Varied according to the	
Available in C Data Security Data is not secured.		,		· ·
16. Data Security 17. Data Visibility modes 18. Declaration Flexibility 18. Declaration Flexibility 19. Variable Name (identifier) length 20. Dealing with uninitialized Variables 19. Variable Name (control flexibility) 20. Dealing with uninitialized Variables 21. Dealing with uninitialized Variables 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (loop) 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays 28. Array Bound Checking 29. Runtime Error handling (Exception Handling) 30. Memory Management 31. Macros 32. Dealors (Supported in C 33. Supported in C 34. Odos not support, a variable must be initialized using a constant secure a visual visual protected in C & Mot supported in C Not supports, a variable sprote a variable spression at the place of destination. 21. Dealing with uninitialized variables in initialized at runtime using entitialized at runtime using initialized at runtime using entitialized at runtime using entitialized at runtime using entitialized at runtime using initialized at runtime using entitialized output. 21. Dealing with uninitialized variable remain undetected in C & thus gives a constant expression. 22. Automatic Coercions 23. Vession Outcome (return value) 24. Iteration statements (loop) 25. Silent Fall-Through in switch statements (loop) 26. Switch applicability 27. Arrays 28. Array Bound Checking 29. Runtime Error handling (Exception Handling) 30. Memory Management 31. Macros 32. Supported in C 33. Supported in C 34. Not supported in C# 35. Versioning support 36. Not supported in C# 37. Not supported in C# 38. Versioning support 38. Not supported in C# 39. Not supported in C# 39. Versioning support 39. Not supported in C# 39. N	15.	Integral Data Type	, , , , , , , , , , , , , , , , , , , ,	C# involves 2 integral types in it.
17. Data Visibility modes C does not provide any data visibility modes visibility mode Declaration Flexibility Inflexible, all variable declarations within a scope occur at the beginning of that scope 19. Variable Name (identifier) length 20. Dynamic Initialization of Variables must be initialized using a constant expression. 21. Dealing with uninitialized Variables undetected in C & thus gives variables with producted in C & thus gives variables with product of destination. 22. Automatic Coercions 23. Test Expression Outcome (return value) 18. Test Expression Outcome (return value) 19. Variable Name (identifier) length 20. Dopring with uninitialized variables remain undetected in C & thus gives variable & gives error messages at completic time to expression at the place of destination. 21. Test Expression Outcome (return value) 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (loop) 24. Iteration statements (loop) 25. Silent Fall-Through in switch statements (loop) for, while, do-while. (loop) for, while, do-while. (loop) for, while, do-while, foreach. 26. Switch applicability Switch can be applied to integers & characters in C in C#, explicit jump statement is required to integers & characters in C in C#, arrays are classes. Arrays are of reference types. Arrays are of value type. 27. Arrays In C, arrays are the collection of elements of the same data type. Arrays are of value type. Array	1.0	Data Cannita		Data is assured
visibility mode		-		
Protected, Internal, Internal Protected.	17.	Data visibility modes		•
18. Declaration Flexibility Inflexible, all variable declarations within a scope occur at the beginning of that scope 19. Variable Name (identifier) length 32 characters No such limit. 20. Dynamic Initialization of Variables Dynamic Initialization Dynamic Initialization of Variables Dynamic Initialization Dynamic Initialization Dynamic Initialization of Variables Dynamic Initialization Dynamic Initialization Dynamic Initialization Dynamic Initialization Dynamic Initialization Dynamic Initialization Dynamic Initialization of Variables Dynamic Initialization Dynamic Initializ			,	
declarations within a scope occur at the beginning of that scope 19. Variable Name (identifier) length 20. Dynamic Initialization of Variables must be initialized using a constant expression. 21. Dealing with uninitialized Uninitialized variable must be initialized using a constant expression. 22. Dealing with uninitialized Uninitialized variable remain undetected in C & thus gives unpredictable output. 22. Automatic Coercions 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (loop) 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays 28. Array Bound Checking 29. Runtime Error handling (Exception Handling) 30. Memory Management 32. Supported in C 33. Macros 34. Supported in C 34. Macros 35. Supported in C 36. Not supported in C# 36. Not supported in C# 37. Mos such limit. 38. No such limit. 39. Not such limit. 30. Not supported in C# 31. Macros 31. Macros 32. Supported in C# 33. Versioning support 32. Variables 33. Variables 34. Versioning support 32. Variables 32. Pointers 33. Variables 32. Variables 32. Variables 33. Variables 34. Variables 36. Variables 36. Variables 37. Variables 37. Variables 38. Variables 38. Variables 39. Variables 39. Variables 39. Variables 39. Variables 30.				
19. Variable Name (identifier) length 32 characters No such limit. 20. Dynamic Initialization of Variables ocosea constant expression. 21. Dealing with uninitialized Variables undetected in C & thus gives unpredictable output. 22. Automatic Coercions Ves, with no warnings if loss could occur could occur could occur constructs return an integer value (0/1) in C 24. Iteration statements (loop) C has 3 iteration statements (loop) for, while, do-while. 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays 28. Arrays 30. Memory Management 31. Macros 32. Robinstructs Supported in C Supported in C# 33. Macros 34. Supported in C 34. Not supported in C# 35. Supported in C 36. Not supported in C# 36. Not supported in C# 37. Macros 38. Supported in C 39. Not supported in C# 39. Pointers 30. Not supported in C# 30. Versioning support 31. Versioning support 32. Variable Namical initialized unstatement support carefulated in c# 34. Versioning support 36. Not supported in C# 37. Not supported in C# 38. Occur at the place of the initialized initialized initialized initialized in initialized initialized in initialized initialized in initialized cursual support initialized unitialized initialized initialized initialized initialized initialized care supported in C# 36. Dealing with uninitialized unities unitialized unities initialized unities initialized unities initialized unities initialized unities initialized unities initialized transitions inities initialized unities initialized unit	18.	Declaration Flexibility		
19. Variable Name (identifier) length 32 characters No such limit.			·	a variable anywhere in the scope.
Dynamic Initialization of Variables must be initialized using a constant expression. 21. Dealing with uninitialized Variables undetected in C & thus gives unpredictable output. 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (loop) 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays Arrays Array Bound Checking An array in C is not bound checked dead. Amanged, using an automatic garbage collector And Supported in C Not supported in C# Not supported in C# Supported in C# Not supported in C# Suppor				
must be initialized using a constant expression. 21. Dealing with uninitialized Uninitialized Uninitialized Variables Uninitialized Uninitialized Variables Uninitialized Uninitialized Variables Uninitialized Variables Uninitialized Variable Rights output. 22. Automatic Coercions Ves, with no warnings if loss could occur Test expression Outcome (return value) 23. Test Expression Outcome (return Test expressions for control flow constructs return an integer value (0/1) in C Value (true/false) in C# Value (true/fals	19.	Variable Name (identifier) length	32 characters	No such limit.
21. Dealing with uninitialized Variables undetected in C & thus gives unpredictable output. 22. Automatic Coercions 23. Test Expression Outcome (return value) 24. Iteration statements (loop) 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays 28. In C, arrays are the collection of elements of the same data type. 29. Array Bound Checking 29. Runtime Error handling (Exception Handling) 30. Memory Management Canton Automatic Coercions Ves, with no warnings if loss could, rust cast explicitly. Test expressions for control flow constructs return an integer value (0/1) in C Chas 3 iteration statements (loop) Chas 3 iteration statements (loop) for, while, do-while. Chas 4 iteration statements (loop) for, while, do-while. Chas 5 iteration statements (loop) for, while, do-while. Chas 6 iteration statements (loop) for, while, do-while, foreach. Chas 6 iteration statements (loop) for, while, do-while, foreach. Chas 6 iteration statements (loop) for, while, do-while, foreach. Chas 6 iteration statements (loop) for, while, do-while, foreach. Chas 6 iteration statements (loop) for do-while, foreach. Chas 6 iteration statements (loop) for do-while, forea	20.	Dynamic Initialization of Variables	1 · · · · · · · · · · · · · · · · · · ·	
Dealing with uninitialized Uninitialized variable remain undetected in C & thus gives unpredictable output. At compile time.				
Variables			constant expression.	· · · · · · · · · · · · · · · · · · ·
Unpredictable output. at compile time.	21.	Dealing with uninitialized	Uninitialized variable remain	C# checks for Uninitialized
22. Automatic Coercions Yes, with no warnings if loss could, must cast explicitly. Test expression Outcome (return value) Test expressions for control flow constructs return an integer value (0/1) in C 24. Iteration statements (loop) C has 3 iteration statements (loop) For, while, do-while, foreach. 25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability Switch applicability Switch can be applied to integers, characters in C in C#, arrays are of value type. Arrays are of value type. Arrays are of value type. 28. Array Bound Checking An array in C is not bound checked C does not formally support Exception handling Exception Handling Memory Management Manual Manual Managed, using an automatic garbage collector 31. Macros Supported in C Not supported in C#		Variables		
could occur 23.	22	Automotic Conning		
Test Expression Outcome (return value) Test expressions for control flow constructs return an integer value (0/1) in C Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow constructs return an integer value (true/false) in C# Test expressions for control flow (log for explicit particles) in C# Test expressions for control flow (log for explicit particles) in C# Test expressions for control flow (log for explicit particles) in C# Test expressions for control flow (log for explicit particles) in C# Test expressions for control flow (log for explicit particles) in C# Test expressions for control flow (log for explicit particles)	22.	Automatic Coercions	_	
value flow constructs return an integer value (0/1) in C	23.	Test Expression Outcome (return		
C has 3 iteration statements (loop) C has 3 iteration statements (loop) for, while, do-while. C# has 4 iteration statements (loop) for, while, do-while, foreach.			flow constructs return an	constructs return an integer
Cloop) for, while, do-while. Cloop) for, while, do-while, foreach.				
Silent Fall-Through in switch statements (free flow case to case)	24.	Iteration statements (loop)		
25. Silent Fall-Through in switch statements (free flow case to case) 26. Switch applicability 27. Arrays Arrays In C, arrays are the collection of elements of the same data type. Arrays are of value type. Arrays are of value type. Arrays are of value type. An array in C is not bound checked checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Macros Suitch can be applied to integers, characters & on string values too in C# An array in C, arrays are the collection of elements of the same data type. An array in C is not bound checked C does not formally support Exception Handling Managed, using an automatic garbage collector 31. Macros Supported in C Not supported in C# 33. Global Variables Supported in C Not supported in C# Supported in C Not supported in C# Supported in C Supported in C Supported in C# Not supported in C#			(100p) for, writte, do-writte.	
Switch applicability Switch can be applied to integers & characters in C 27. Arrays In C, arrays are the collection of elements of the same data type. Array Bound Checking Array Bound Checking Runtime Error handling (Exception Handling) 30. Memory Management Manual Macros Supported in C Switch can be applied to integers, characters & on string values too in C# In C#, arrays are classes. Arrays are of reference types. An array in C# is bound checked C# Supports Exception handling C# Supports Exception handling Managed, using an automatic garbage collector 31. Macros Supported in C Not supported in C# 32. Pointers Supported in C Supported in C Not supported in C# Typedef statement Supported in C Not supported in C# Supported in C Not supported in C# Supported in C Not supported in C# Supported in C Supported in C# Supported in C Supported in C# Supported in C Supported in C#	25.	_	Allowed in C	
integers & characters in C 27. Arrays In C, arrays are the collection of elements of the same data type. Arrays are of value type. Arrays are of value type. 28. Array Bound Checking An array in C is not bound checked checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Managed, using an automatic garbage collector 31. Macros Supported in C Supported in C Not supported in C# 32. Pointers Supported in C Supported in C Not supported in C# 34. Typedef statement Supported in C Supported in C Supported in C Supported in C Not supported in C# Supported in C Not supported in C# Supported in C Supported in C Supported in C Supported in C Supported in C# Supported in C Supported in C# Supported in C Supported in C#				
in C# 27. Arrays In C, arrays are the collection of elements of the same data type. Arrays are of value type. 28. Array Bound Checking An array in C is not bound checked checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Macros Supported in C 31. Macros Supported in C 32. Pointers Supported in C Supported in C Mot supported in C# Not supported in C# Supported in C Not supported in C# Supported in C# Supported in C Not supported in C#	26.	Switch applicability		
27. Arrays In C, arrays are the collection of elements of the same data type. Arrays are of value type. 28. Array Bound Checking An array in C is not bound checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Managed, using an automatic garbage collector 31. Macros 32. Pointers Supported in C Supported in C Supported in C Supported in C Not supported in C# 33. Global Variables Supported in C Supported in C Not supported in C# Supported in C Supported in C# Supported in C# Supported in C# Supported in C Supported in C#			integers & characters in C	_
elements of the same data type. Arrays are of value type. An array in C is not bound checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Macros Supported in C 31. Macros Supported in C Not supported in C# 31. Typedef statement Supported in C Supported in C Not supported in C# Supported in C Supported in C# Supported in C Supported in C# Supported in C Supported in C#	27.	Arrays	In C, arrays are the collection of	
Arrays are of value type. 28. Array Bound Checking An array in C is not bound checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Managed, using an automatic garbage collector 31. Macros 32. Pointers Supported in C 33. Global Variables Supported in C Supported in C Supported in C Not supported in C# Supported in C Not supported in C# Not supported in C# Supported in C Supported in C# Supported in C Supported in C#			elements of the same data	
Array Bound Checking An array in C is not bound checked 29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Macros Supported in C Supported in C Supported in C Supported in C Mot supported in C# Supported in C# Supported in C Mot supported in C# Supported in C Not supported in C# Supported in C Supported in C Supported in C Supported in C# Supported in C Supported in C#			7.7	
checked 29. Runtime Error handling (Exception Handling) 30. Memory Management 31. Macros 32. Pointers 33. Global Variables 34. Typedef statement Supported in C Not supported in C# Not supported in C# Supported in C Supported in C# Supported in C Supported in C#	28	Array Bound Checking		An array in C# is hound checked
29. Runtime Error handling (Exception Handling) 30. Memory Management Manual Manual Manual Manual Manual Mot supported in C Supported in C Supported in C Mot supported in C# Supported in C Mot supported in C# Supported in C Not supported in C# Supported in C Not supported in C# Not supported in C# Supported in C Not supported in C# Supported in C# Supported in C Supported in C# Supported in C# Supported in C# Supported in C#	20.	ay bound effecting	·	An array in on 13 bound checked
30. Memory Management Manual Managed, using an automatic garbage collector 31. Macros Supported in C Not supported in C# 32. Pointers Supported in C Not supported in C# 33. Global Variables Supported in C Not supported in C# 34. Typedef statement Supported in C Not supported in C# 35. Versioning support Not supported in C Supported in C#	29.	Runtime Error handling (Exception	C does not formally support	C# Supports Exception handling
garbage collector 31. Macros Supported in C Not supported in C# 32. Pointers Supported in C Not supported in C# 33. Global Variables Supported in C Not supported in C# 34. Typedef statement Supported in C Not supported in C# 35. Versioning support Not supported in C Supported in C#				
31. Macros Supported in C Not supported in C# 32. Pointers Supported in C Not supported in C# 33. Global Variables Supported in C Not supported in C# 34. Typedef statement Supported in C Not supported in C# 35. Versioning support Not supported in C Supported in C#	30.	Memory Management	Manual	
32. Pointers Supported in C Not supported in C# 33. Global Variables Supported in C Not supported in C# 34. Typedef statement Supported in C Not supported in C# 35. Versioning support Not supported in C Supported in C#	31	Macros	Supported in C	
33. Global Variables Supported in C Not supported in C# 34. Typedef statement Supported in C Not supported in C# 35. Versioning support Not supported in C Supported in C#				
35. Versioning support Not supported in C Supported in C#				• •
36. Support for metadata Not supported in C Supported in C#				
	36.	Support for metadata	Not supported in C	Supported in C#