|  |  |
| --- | --- |
| **Code** | **Quadruple** |
| int x;  int y = 5;  int z = x + y;  z = x - y;  z = x \* y;  z = x / y;  z = x % y;  x++;  x--;  z = (x + y);  exit  // Errors:  int x;  int y = 5;  int z = x + y  z = x - y;  z = x \* y;  z = x / y;  z = x % y;  x++;  x--;  z = (x + y;  exit | OpCode: 0 Arg1: Arg2: Result: x  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 5 Arg2: Result: y  OpCode: 10 Arg1: x Arg2: y Result: Temp1  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 1 Arg1: Temp1 Arg2: Result: z  OpCode: 11 Arg1: x Arg2: y Result: Temp1  OpCode: 1 Arg1: Temp1 Arg2: Result: z  OpCode: 12 Arg1: x Arg2: y Result: Temp1  OpCode: 1 Arg1: Temp1 Arg2: Result: z  OpCode: 13 Arg1: x Arg2: y Result: Temp1  OpCode: 1 Arg1: Temp1 Arg2: Result: z  OpCode: 14 Arg1: x Arg2: y Result: Temp1  OpCode: 1 Arg1: Temp1 Arg2: Result: z  OpCode: 15 Arg1: INC Arg2: INC Result: x  OpCode: 16 Arg1: DEC Arg2: DEC Result: x  OpCode: 10 Arg1: x Arg2: y Result: Temp1  OpCode: 1 Arg1: Temp1 Arg2: Result: z |
| int x = 4;  if (x == 5) { x = 6; int y = 7;}  if ( x == 4) { x = 6; int z = 7;} else { x = 7; int y = 8;}  if ( x == 3) { x = 4; } else if (true) { x = 8;} else {x = 3;}  int y = 3;  if (3 > 4) { int z = 3;}  // Warning for always false  // Notice our compiler handles comments :D  exit  // Error:  int x = 4;  if (x == 5) { x = 6; int y = 7;}  if ( x == 4) { x = 6; int z = 7;} else { x = 7; int y = 8;}  if ( x == 3) { x = 4; } else if (true) { x = 8;} else {x = 3;}  int 3y = 3;  if (3 > 4) { int z = 3;}  // Warning for always false  // Notice our compiler handles comments :D  exit | OpCode: 0 Arg1: Arg2: Result: x  OpCode: 1 Arg1: 4 Arg2: Result: x  OpCode: 33 Arg1: x Arg2: 5 Result: Temp1  OpCode: 60 Arg1: IF Arg2: OpenIf Result: 0  OpCode: 1 Arg1: 6 Arg2: Result: x  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 7 Arg2: Result: y  OpCode: 60 Arg1: IF Arg2: CloseIf Result: 0  OpCode: 33 Arg1: x Arg2: 4 Result: Temp1  OpCode: 60 Arg1: IF Arg2: OpenIf Result: 1  OpCode: 1 Arg1: 6 Arg2: Result: x  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 1 Arg1: 7 Arg2: Result: z  OpCode: 80 Arg1: else Arg2: 0 Result: 1  OpCode: 1 Arg1: 7 Arg2: Result: x  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 8 Arg2: Result: y  OpCode: 81 Arg1: 0 Arg2: CloseElse Result:  OpCode: 33 Arg1: x Arg2: 3 Result: Temp1  OpCode: 60 Arg1: IF Arg2: OpenIf Result: 2  OpCode: 1 Arg1: 4 Arg2: Result: x  OpCode: 60 Arg1: 1 Arg2: OpenElseIf1 Result: 2  OpCode: 60 Arg1: IF Arg2: OpenElseIf2 Result: 3  OpCode: 1 Arg1: 8 Arg2: Result: x  OpCode: 80 Arg1: else Arg2: 1 Result: 3  OpCode: 1 Arg1: 3 Arg2: Result: x  OpCode: 81 Arg1: 1 Arg2: CloseElse Result:  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 3 Arg2: Result: y  OpCode: 28 Arg1: 3 Arg2: 4 Result: Temp1  OpCode: 60 Arg1: IF Arg2: OpenIf Result: 4  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 1 Arg1: 3 Arg2: Result: z  OpCode: 60 Arg1: IF Arg2: CloseIf Result: 4 |
| while ( true ) { int x = 4; int y = x; float z = 3.4;  float u = 3;}  char y = 'a';  repeat { int x = 6; y = 'b';  x++;  }  until ( 3 > 4 );  # Warning  // whoops, I should use // for comments not #  exit | OpCode: 20 Arg1: 0 Arg2: 0 Result: OpenWhile1  OpCode: 20 Arg1: 0 Arg2: 0 Result: OpenWhile2  OpCode: 0 Arg1: Arg2: Result: x  OpCode: 1 Arg1: 4 Arg2: Result: x  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: x Arg2: Result: y  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 1 Arg1: 3.400000 Arg2: Result: z  OpCode: 0 Arg1: Arg2: Result: u  OpCode: 1 Arg1: 3.000000 Arg2: Result: u  OpCode: 90 Arg1: 0 Arg2: Result: CloseWhile  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 'a' Arg2: Result: y  OpCode: 22 Arg1: 0 Arg2: 0 Result: OpenrepeatUntil  OpCode: 0 Arg1: Arg2: Result: x  OpCode: 1 Arg1: 6 Arg2: Result: x  OpCode: 1 Arg1: 'b' Arg2: Result: y  OpCode: 15 Arg1: INC Arg2: INC Result: x  OpCode: 28 Arg1: 3 Arg2: 4 Result: Temp1  OpCode: 91 Arg1: 0 Arg2: Result: CloseRepeatUntil |
| for (int i = 2; i < 3; i++ ) { string x = "Hello World";}  // Error:  for (int i = 2; i < 3; i++ ) { string x = "Hello World";}  int x=;  exit | OpCode: 1 Arg1: 2 Arg2: Result: i  OpCode: 21 Arg1: 0 Arg2: 0 Result: OpenForLoop1  OpCode: 29 Arg1: i Arg2: 3 Result: Temp1  OpCode: 21 Arg1: 0 Arg2: 0 Result: OpenForLoop2  OpCode: 15 Arg1: INC Arg2: INC Result: i  OpCode: 0 Arg1: Arg2: Result: x  OpCode: 1 Arg1: "Hello World" Arg2: Result: x  OpCode: 92 Arg1: 0 Arg2: Result: CloseForLoop |
| int y = 0;  int x = 3;  switch (x) { case 1: y = 1; break;  case 2: y = 2; break; default: y = 3;}  exit  // Error:  int y = 0;  int x = 3;  switch (x) { case 1: y = 1; break;  case a2: y = 2; break; default: y = 3;}  exit | OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 0 Arg2: Result: y  OpCode: 0 Arg1: Arg2: Result: x  OpCode: 1 Arg1: 3 Arg2: Result: x  OpCode: 61 Arg1: SwitchStart Arg2: Result: x  OpCode: 70 Arg1: 1 Arg2: Result: x  OpCode: 70 Arg1: caseSecondTime Arg2: 1 Result: x  OpCode: 1 Arg1: 1 Arg2: Result: y  OpCode: 70 Arg1: caseThirdTime Arg2: 1 Result: 0  OpCode: 70 Arg1: 2 Arg2: Result: x  OpCode: 70 Arg1: caseSecondTime Arg2: 2 Result: x  OpCode: 1 Arg1: 2 Arg2: Result: y  OpCode: 70 Arg1: caseThirdTime Arg2: 2 Result: 0  OpCode: 71 Arg1: 0 Arg2: Result: DefaultCase  OpCode: 1 Arg1: 3 Arg2: Result: y  OpCode: 72 Arg1: 0 Arg2: ENDSWITCH Result: |
| enum x { a = 0, b = 2, c = 4};  int y = a;  y = b;  // Error:  enum x { a = 0, b = 2, c = 4};  int y = a;  y = b;  intz =;  // intz;  exit | OpCode: 23 Arg1: 0 Arg2: 0 Result: OpenEnum  OpCode: 0 Arg1: Arg2: Result: c  OpCode: 1 Arg1: 4 Arg2: Result: c  OpCode: 0 Arg1: Arg2: Result: b  OpCode: 1 Arg1: 2 Arg2: Result: b  OpCode: 0 Arg1: Arg2: Result: a  OpCode: 1 Arg1: 0 Arg2: Result: a  OpCode: 73 Arg1: 0 Arg2: ENDENUM Result:  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: a Arg2: Result: y  OpCode: 1 Arg1: b Arg2: Result: y |
| int Func (int y, int z) {int sum = y + z; return sum;}  int Func2 () {int y = 3; int z = 4; int sum = y + z; return sum;}  void Func3 (int y, int z) {y = 5; z = 6; return;}  void Func4 (int y, int z) {y = 5;}  void Func5 () {int y = 5; int z = 6; return;}  int a = 3;  int b = 4;  Func (a, b);  int z = Func2();  a = Func2();  # OpCode: 100 Arg1: 1 Arg2: 6 Result: Func6  OpCode 100 ---> opening function  Arg1: 1 ---> scope number  Arg2: 6 ---> number of arguments  Result: Func6 ---> Function identifier  // Error:  int Func (int y, int z) {int sum = y + z; return sum;}  int Func2 () {int y = 3; int z = 4; int sum = y + z; return sum;}  void Func3 (int y, int z) {y = 5; z = 6; return;}  void Func4 (int y, int z) {y = 5;}  void Func5 () {int y = 5; int z = 6; return;}  int a = 3;  int b = 4;  Func (a, b);  int z = Func2();  a = Func2();  b = Func4(;  exit | OpCode: 0 Arg1: Arg2: Result: z  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 100 Arg1: 1 Arg2: 1 Result: Func  OpCode: 10 Arg1: y Arg2: z Result: Temp1  OpCode: 0 Arg1: Arg2: Result: sum  OpCode: 1 Arg1: Temp1 Arg2: Result: sum  OpCode: 101 Arg1: ret Arg2: 1 Result:  OpCode: 100 Arg1: 1 Arg2: 2 Result: Func2  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 3 Arg2: Result: y  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 1 Arg1: 4 Arg2: Result: z  OpCode: 10 Arg1: y Arg2: z Result: Temp1  OpCode: 0 Arg1: Arg2: Result: sum  OpCode: 1 Arg1: Temp1 Arg2: Result: sum  OpCode: 101 Arg1: ret Arg2: 2 Result:  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 100 Arg1: 1 Arg2: 3 Result: Func3  OpCode: 1 Arg1: 5 Arg2: Result: y  OpCode: 1 Arg1: 6 Arg2: Result: z  OpCode: 101 Arg1: ret Arg2: 3 Result:  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 100 Arg1: 1 Arg2: 4 Result: Func4  OpCode: 1 Arg1: 5 Arg2: Result: y  OpCode: 101 Arg1: ret Arg2: 4 Result:  OpCode: 100 Arg1: 1 Arg2: 5 Result: Func5  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 1 Arg1: 5 Arg2: Result: y  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 1 Arg1: 6 Arg2: Result: z  OpCode: 101 Arg1: ret Arg2: 5 Result:  OpCode: 0 Arg1: Arg2: Result: z2  OpCode: 0 Arg1: Arg2: Result: y2  OpCode: 0 Arg1: Arg2: Result: x2  OpCode: 0 Arg1: Arg2: Result: z  OpCode: 0 Arg1: Arg2: Result: y  OpCode: 0 Arg1: Arg2: Result: x  OpCode: 100 Arg1: 1 Arg2: 6 Result: Func6  OpCode: 1 Arg1: 3 Arg2: Result: y  OpCode: 101 Arg1: ret Arg2: 6 Result:  OpCode: 0 Arg1: Arg2: Result: a  OpCode: 1 Arg1: 3 Arg2: Result: a  OpCode: 0 Arg1: Arg2: Result: b  OpCode: 1 Arg1: 4 Arg2: Result: b  OpCode: 63 Arg1: Arg2: Func Result: FunctionCall  OpCode: 63 Arg1: Arg2: Func2 Result: z  OpCode: 63 Arg1: Arg2: Func2 Result: a |