2.2 Variables (int)

Variable declarations

A ***variable declaration*** is a statement that declares a new variable, specifying the variable's name and type. Ex: `int userAge;` declares a new variable named userAge that can hold an integer value. The compiler allocates a memory location for userAge capable of storing an integer. Ex: In the animation below, the compiler allocated userAge to memory location 97, which is known as the variable's address. The choice of 97 is arbitrary and irrelevant to the programmer, but the idea that a variable corresponds to a memory location is important to understand.

When a statement that assigns a variable with a value executes, the processor writes the value into the variable's memory location. Likewise, reading a variable's value reads the value from the variable's memory location. The programmer must declare a variable before any statement that assigns or reads the variable, so that the variable's memory location is known.

PARTICIPATION ACTIVITY 2.2.1: A variable refers to a memory location.

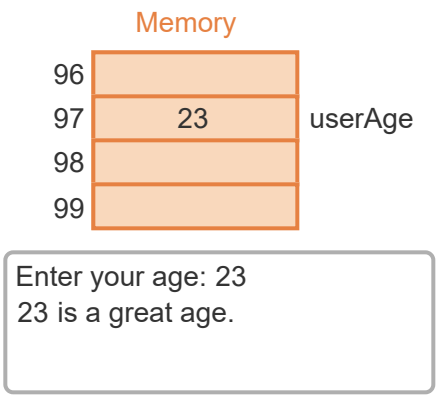
1 2 3 4 2x speed

```
#include <iostream>
using namespace std;

int main() {
   int userAge;

   cout << "Enter your age: ";
   cin >> userAge;
   cout << userAge << " is a great age." << endl;

   return 0;
}
```



cout prints userAge's value to screen.

Feedback?

PARTICIPATION ACTIVITY 2.2.2: Declaring integer variables.

Note: Capitalization matters, so MyNumber is not the same as myNumber.

1) Declare an integer variable named numPeople. (Do not initialize the variable.)

```
int numPeople;
```

Check Show answer

Correct

```
int numPeople;
```

The compiler will allocate a particular memory location for numPeople.

2) Using two statements on separate lines, declare integer variables named newSales and totalSales. (Do not initialize the variables.)

```
int newSales;
int totalSales;
```

Check Show answer

Correct

```
int newSales;
int totalSales;
```

Multiple variables can be declared in the same statement, as in: `int newSales, totalSales;`, but this material usually avoids that style.

3) What memory location (address) will a compiler allocate for the variable declaration below. If appropriate, type: Unknown

```
int numHouses = 99;
```

Unknown

Check Show answer

Correct

```
Unknown
```

A programmer does not know the specific location, but understanding that a specific location is allocated for a variable is important.

Feedback?

Compiler optimization

Modern compilers may optimize variables away, allocate variables on the stack, or use registers for variables. However, the conceptual view of a variable in memory helps understand many language aspects.

Assignment statements

An ***assignment statement*** assigns the variable on the left-side of the = with the current value of the right-side expression. Ex: `numApples = 8;` assigns numApples with the value of the right-side expression (in this case 8). assign

An ***expression*** may be a number like 80, a variable name like numApples, or a simple calculation like numApples + 1. Simple calculations can involve standard math operators like +, -, and *, and parentheses as in 2 * (numApples - 1). An integer like 80 appearing in an expression is known as an ***integer literal***.

In the code below, litterSize is assigned with 3, and yearlyLitters is assigned with 5. Later, annualMice is assigned with the value of litterSize * yearlyLitters (3 * 5, or 15), which is then printed. Next, litterSize is assigned with 14, yearlyLitters is assigned with 10, and annualMice is assigned with their product (14 * 10, or 140), which is printed.

Figure 2.2.1: Assigning a variable.

```
#include <iostream>
using namespace std;

int main() {
   int litterSize;
   int yearlyLitters;
   int annualMice;

   litterSize    = 3; // Low end of litter size
range
   yearlyLitters = 5; // Low end of litters per
year

   cout << "One female mouse may give birth to ";
   annualMice = litterSize * yearlyLitters;
   cout << annualMice << " mice," << endl;

   litterSize    = 14; // High end
   yearlyLitters = 10; // High end

   cout << "and up to ";
   annualMice = litterSize * yearlyLitters;
   cout << annualMice << " mice, in a year." <<
endl;

   return 0;
}
```

```
One female mouse may give birth to 15
mice,
and up to 140 mice, in a year.
```

Feedback?

PARTICIPATION ACTIVITY 2.2.3: Assignment statements.

Be sure to end assignment statements with a semicolon (;).

1) Write an assignment statement to assign numCars with 99.

numCars = 99;

Check Show answer

Correct

numCars = 99;

The statement assigns the variable numCars with the value 99.

2) Assign houseSize with 2300.

houseSize = 2300;

Check Show answer

Correct

houseSize = 2300;

The program assigns the variable houseSize with the value 2300.

3) Assign numFruit with the current value of numApples.

numFruit = numApples;

Check Show answer

Correct

numFruit = numApples;

The program evaluates the value of numApples by reading the value held in numApples' memory location, and then assigns numFruit with that value.

4) The current value in houseRats is 200. What is in houseRats after executing the statement below? Valid answers: 0, 199, 200, or unknown.

```
numRodents = houseRats;
```

200

Check Show answer

Correct

200

The statement evaluates the value of houseRats by reading the value held in houseRats' memory location, and stores a copy of that value into numRodents. houseRats doesn't change.

5) Assign numItems with the result of ballCount - 3.

numItems = ballCount - 3;

Check Show answer

Correct

numItems = ballCount - 3;

The program reads the value of ballCount, subtract 3 from that value, and assigns numItems with the result.

6) dogCount is 5. What is in animalsTotal after executing the statement below?

```
animalsTotal = dogCount - 3;
```

Correct

2

2

Check Show answer

The statement reads the value of dogCount (5), subtracts 3, and assigns animalsTotal with the result of 2.

7) dogCount is 5. What is in dogCount after executing the statement below?

```
animalsTotal = dogCount - 3;
```

5

Check Show answer

Correct

5

The statement reads the value of dogCount, but does not change the value of dogCount.

8) What is in numBooks after both statements execute?

```
numBooks = 5;
numBooks = 3;
```

3

Check Show answer

Correct

3

The first statement assigns numBooks with 5. The second statement assigns numBooks with 3, replacing the previously held value 5.

Feedback?

CHALLENGE ACTIVITY 2.2.1: Enter the output of the variable assignments.

Jump to level 1

Type the program's output.

```
#include <iostream>
using namespace std;

int main() {
   int x;
   int y;

   y = 5;
   x = y + 4;
   y = 12;

   cout << x << " " << y;

   return 0;
}
```

9 12

1 2 3 4

Check Next **Done**. Click any level to practice more. Completion is preserv

✓ x is assigned with the value of y plus 4, then y is assigned with 12.

Yours `9 12`

Expected `9 12`

Feedback?

CHALLENGE ACTIVITY 2.2.2: Assigning a sum.

Write a statement that assigns numCoins with numNickels + numDimes. Ex: 5 nickels and 6 dimes results in 11 coins.

Note: These activities may test code with different test values. This activity will perform two tests: the first with nickels = 5 and dimes = 6, the second with nickels = 9 and dimes = 0. See How to Use zyBooks.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5     int numCoins;
6     int numNickels;
7     int numDimes;
8
9     numNickels = 5;
10    numDimes = 6;
11
12    /* Your solution goes here  */
13    numCoins = numNickels + numDimes;
14
15    cout << "There are  " << numCoins << " coins" << endl;
16
17    return 0;
18 }
```

Run ✓ All tests passed

✓ Testing with nickels = 5 and dimes = 6

Your value 11

✓ Testing with nickels = 9 and dimes = 0

Your value 9

Feedback?

Initializing variables

Although not required, an integer variable is often assigned an initial value when declared. Ex: `int maxScore = 100;` declares an int variable named maxScore with an initial value of 100.

Figure 2.2.2: Variable initialization: Example program.

```
#include <iostream>
using namespace std;

int main() {
   int avgLifespan = 70;
   int userAge;

   cout << "Enter your age: ";
   cin >> userAge;
   cout << userAge << " is a great age" << endl;

   cout << "Average lifespan is " << avgLifespan << endl;

   return 0;
}
```

```
Enter your age: 24
24 is a great age
Average lifespan is 70
```

Feedback?

PARTICIPATION ACTIVITY 2.2.4: Declaring and initializing integer variables.

1) Declare an integer variable named numDogs, initializing the variable to 0 in the declaration.

int numDogs = 0;

Check Show answer

Correct

int numDogs = 0;

The compiler will allocate a particular memory location for numDogs, and initially store 0 in that location.

2) Declare an integer variable named daysCount, initializing the variable to 365 in the declaration.

int daysCount = 365;

Check Show answer

Correct

int daysCount = 365;

The compiler will allocate a particular memory location for daysCount, and initially store 365 in that location.

Feedback?

CHALLENGE ACTIVITY 2.2.3: Declaring and initializing variables.

Write one statement that declares an integer variable numHouses initialized to 25.

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5
6  int numHouses = 25;
7     /* Your solution goes here  */
8
9     cout << numHouses << endl;
10
11    return 0;
12 }
```

Run ✓ All tests passed

✓ Testing for numHouses value

Your value 25

Feedback?

Assignment statement with same variable on both sides

Commonly, a variable appears on both the right and left side of the = operator. Ex: If numItems is 5, after `numItems = numItems + 1` executes, numItems will be 6. The statement reads the value of numItems (5), adds 1, and assigns numItems with the result of 6, which replaces the value previously held in numItems.

PARTICIPATION ACTIVITY | 2.2.5: Variable assignments overwrite a variable's previous values: People-known example.

1 2 3 4 5 6 7 8 2x speed

```
#include <iostream>
using namespace std;

int main() {
   int yourFriends;
   int totalFriends;

   cout << "Enter the number of people you know: ";
   cin >> yourFriends;
   totalFriends = yourFriends;
   cout << " You know " << totalFriends << " people.\n";
   totalFriends = totalFriends * yourFriends;
   cout << " Those people know " << totalFriends << " people.\n";
   totalFriends = totalFriends * yourFriends;
   cout << " And they know " << totalFriends << " people.\n\n";

   return 0;
}
```

```
Enter the number of people you know:  200
You know 200 people.
Those people know 40000 people.
And they know 8000000 people.
```

96	??	
97	200	yourFriends
98	8000000	totalFriends
99	??	

Assignment reads totalFriends (now 40000) and yourFriends (200), multiplies those values, and assigns totalFriends with the result of 8000000.

Feedback?

Six degrees of separation

The above example relates to the popular idea that any two people on earth are connected by just "six degrees of separation", accounting for overlapping of known-people.

PARTICIPATION ACTIVITY 2.2.6: Assignment statements with same variable on both sides.

1) numApples is initially 5. What is numApples after:

```
numApples = numApples + 3;
```

8

Check Show answer

Correct

8

The statement reads the current value of numApples (5), adds 3, and assign numApples with the result of 8.

2) numApples is initially 5. What is numFruit after:

```
numFruit  = numApples;
numFruit  = numFruit + 1;
```

6

Check Show answer

Correct

6

The first statement assigns numFruit with 5. The second statement reads numFruit (5), adds 1, and assigns numFruit with the result of 6.

3) Write a statement ending with -1 that decreases variable flyCount's value by 1.

flyCount = flyCount -1;

Check Show answer

Correct

flyCount = flyCount - 1;

The statement reads the value of flyCount, subtracts 1, and assigns flyCount with the result (overwriting the previous value).

Feedback?

CHALLENGE ACTIVITY 2.2.4: Adding a number to a variable.

Write a statement that increases numPeople by 5. Ex: If numPeople is initially 10, the output is: There are 15 people.

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5    int numPeople;
6
```

```
   cin >> numPeople;

   /* Your solution goes here  */
   numPeople = numPeople + 5;

   cout << "There are " << numPeople << " people." << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with numPeople initially 10

Your output `There are 15 people.`

✓ Testing with numPeople initially 99

Your output `There are 104 people.`

✓ Testing with numPeople initially 0

Your output `There are 5 people.`

Feedback?

Common errors

A common error is to read a variable that has not yet been assigned a value. If a variable is declared but not initialized, the variable's memory location contains some unknown value, commonly but not always 0. A program with an uninitialized variable may thus run correctly on system that has 0 in the memory location, but then fail on a different system—a very difficult bug to fix. A programmer must ensure that a program assigns a variable with a value before reading.

A common error by new programmers is to write an assignment statement in reverse. Ex: `numKids + numAdults = numPeople`, or `9 = beansCount`. Those statements won't compile, but writing `numCats = numDogs` in reverse *will* compile, leading to a hard-to-find bug.

PARTICIPATION ACTIVITY 2.2.7: Common errors.

Which code segments have an error?

1)
```
21 = dogCount;
```

- (•) Error
- ( ) No error

Correct

The statement will not compile because the left side must be a variable. The programmer likely meant to write `dogCount = 21;`

2)
```
int amountOwed = -999;
```

- ( ) Error
- (•) No error

Correct

A variable can be initialized when declared, and an int variable can hold negative values.

3)
```
int numDays;
int numYears;

numDays = numYears * 365;
```

- (•) Error
- ( ) No error

Correct

numYears is used without having been assigned with a value, so the value of numYears is unknown. Most modern compilers check if a variable is used before being assigned with a value, reporting an error or warning.

Feedback?

(*assign) We ask instructors to give us leeway to teach the idea of an "assignment statement," rather than the language's actual "assignment expression," whose use we condone primarily in a simple statement.