

OUTLINE

1 CHAPTER 8: A C++ INTRODUCTION FOR PYTHON PROGRAMMERS

- Decision Statements
- Type Conversions
- Looping Statements
- Arrays
- In-Class work

IF STATEMENT

INDENTATION IS MEANINGLESS

Since C++ ignores indentation, multiple statements in an `if` statement must be enclosed in braces to form a block.

C++ HAS NO ELIF

Multiple conditional branches in C++ require nested if statements.

See programs `if1.cpp`, `if2.cpp`, `if3.cpp`, `grades.cpp`

IF STATEMENT

INDENTATION IS MEANINGLESS

Since C++ ignores indentation, multiple statements in an `if` statement must be enclosed in braces to form a block.

C++ HAS NO `ELIF`

Multiple conditional branches in C++ require nested if statements.

See programs [if1.cpp](#), [if2.cpp](#), [if3.cpp](#), [grades.cpp](#)

DATA TYPES

IMPLICIT CONVERSION

Like Python, C++ automatically converts ints to floats or doubles when adding an integer to a float value.

EXPLICIT CONVERSION

C++ also performs explicit conversion with the `static_cast` keyword.

See program `typeConvs.cpp`

DATA TYPES

IMPLICIT CONVERSION

Like Python, C++ automatically converts ints to floats or doubles when adding an integer to a float value.

EXPLICIT CONVERSION

C++ also performs explicit conversion with the `static_cast` keyword.

See program [typeConvs.cpp](#)

LOOPS

FOR LOOPS OVER DIFFERENT VALUES OF A SINGLE VARIABLE

In Python, a for loop must use a pre-existing sequence of values, like the list returned by the function `range`.

In C++, a variable changes its value by incrementing or decrementing its value for each iteration of the loop.

```
int i;  
for (i=0;i<10;i++)  
{  
    cout << i << endl;  
}
```

See a slightly different example: [loopExample1.cpp](#)

LOOPS

WHILE LOOPS

While loops in C++ are exactly like those in Python. They test for a boolean value to be true before they begin each iteration of the loop. When it is false, the loop is exited.

```
int i;  
while i < 10  
{  
    cout << i << endl;  
    i += 1; }
```

LOOPS

DO-WHILE LOOPS

Do-while loops in C++ are different than while loops in Python. They only test for the boolean value to be true at the end of each loop iteration. This means that a loop is guaranteed to execute at least once.

```
int i=0;
do {
    cout << i << endl;
    i++;
} while i<10;
```


SINGLE-DIMENSION ARRAYS

NOT AS SAFE AS PYTHON LISTS

- No range checking is performed when an index is used.
- Can initialize values when declaring an array.

```
int i,n;

cout << "Enter size of the array:";
cin >> n;

int a[n];
for (i=0 ; i<n ; i++) {
    a[i] = i+1;
}
```

MULTI-DIMENSIONAL ARRAYS

ARRAYS OF ARRAYS

- Any number of dimensions is supported.
- Consecutive addresses are easily found (fast random access via a formula).

```
int i,j,n,m;

cout << "Enter the number of rows:"; cin >> n;
cout << "Enter the number of columns:"; cin >> m;

int a[n][m];
for(i=0 ; i<n ; i++) {
    for(i=0 ; i<n ; i++) {
        a[i][j] = i+1;
    }
}
```

ARRAYS OF CHARACTERS

USING ARRAYS OF CHARACTERS IS RISKY

```
char c[20];  
cout << "enter your first name: ";  
  
// this code is a security risk  
// a buffer overflow occurs if the user enters  
// more than 19 characters  
cin >> c;  
  
cout << "Hello " << c << endl;
```

ARRAYS OF CHARACTERS

C STRINGS

- char arrays with terminating zero in last position.
- Literal string values (in quotes) are zero-terminated strings in C++.
- Convenient library functions for concatenation (`strcat`) etc.
- `#include <string.h>` for library functions

ARRAYS OF CHARACTERS

BETTER TO USE THE C++ STRING CLASS.

- `#include <string>`
- An object's capacity increases automatically.
- Convenient member functions for concatenation and other operations.
- Easy to initialize a C++ string from a char array.

See programs [buffer.cpp](#), [str1.cpp](#)

ARRAYS OF NUMBERS

Build the following matrix of values and display it.

$$\begin{array}{c} \text{rows} \\ n \end{array} \begin{array}{c} \text{columns} \\ m \end{array} \left[\begin{array}{ccccc} a_{11} & a_{12} & a_{13} & \dots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2m} \\ \dots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nm} \end{array} \right]$$

where
 $a_{ij} = (i+j)^2$
 (i stands for row,
 j stands for column)

n and m are provided by the user.

ARRAYS OF NUMBERS

For example, for $n=2$ and $m=3$ (2 rows and 3 columns) the matrix will be:

$$A_{2 \times 3} = \begin{bmatrix} 4 & 9 & 16 \\ 9 & 16 & 25 \end{bmatrix}$$