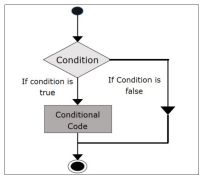
CSE 101
Slide Set 6

Doç. Dr. Mehmet Göktürk
Department of Computer Engineering

1

Conditional Expressions

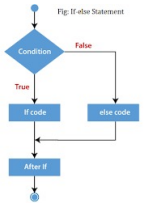
```
if( ............ ){
    bla bla bla
    bla bla bla
}
```



2

Conditional Expressions

```
if( ............ ){
    bla bla bla
    bla bla bla
}
else{
    bla bla bla
    bla bla bla
}
```



3

Conditional Cascaded

```
if( ............ ){
    bla bla bla
    if( ............ ){
        bom bom
    }
    blu blu blu
}
else{
    if ( ............ ){
        bla bla bla
        bla bla blom
    }
    else{
        bom
    }
    bambum
}
```


4

Conditional Cascaded

```
if( ............. ){
    bla bla bla
    if( ............. ){
        bom bom
    }
    blu blu blu
}
else if( ............. ){
        bla bla bla
        bla bla blom
}
else{
    bom
}
```

bambum?

5

Line Counting

```
#include <stdio.h>

/* count lines in input */
main()
{
    int c, nl;

    nl = 0;
    while ((c = getchar()) != EOF)
        if (c == '\n')
            ++nl;
    printf("%d\n", nl);
}
```


6

Word Counting

```
#include <stdio.h>

#define IN  1    /* inside a word */
#define OUT 0    /* outside a word */

/* count lines, words, and characters in input */
main()
{
    int c, nl, nw, nc, state;

    state = OUT;
    nl = nw = nc = 0;
    while ((c = getchar()) != EOF) {
        ++nc;
        if (c == '\n')
            ++nl;
        if (c == ' ' || c == '\n' || c == '\t')
            state = OUT;
        else if (state == OUT) {
            state = IN;
            ++nw;
        }
    }
    printf("%d %d %d\n", nl, nw, nc);
}
```


7

Pay attention to:

```
nl = nw = nc = 0;
```

```
nl = (nw = (nc = 0));
```

```
if (c == ' ' || c == '\n' || c == '\t')
```


8

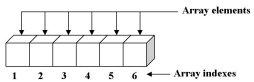
Examples..

9

10

Arrays

- Write a program to count the number of occurrences of each digit, of white space characters (blank, tab, newline), and of all other characters.



One-dimensional array with six elements

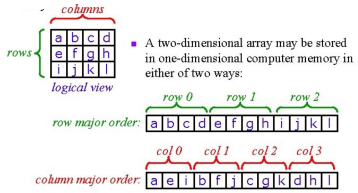
11

Two dimensional

	Col1	Col2	Col3	Col4	....
Row1	Arr[0][0]	Arr[0][1]	Arr[0][2]	Arr[0][3]	
Row2	Arr[1][0]	Arr[1][1]	Arr[1][2]	Arr[1][3]	
Row3	Arr[2][0]	Arr[2][1]	Arr[2][2]	Arr[2][3]	
Row4	Arr[3][0]	Arr[3][1]	Arr[3][2]	Arr[3][3]	
. . .					

12

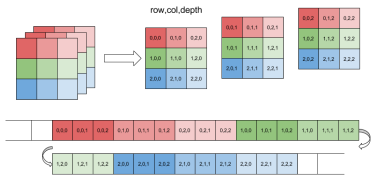
Memory Allocation of Arrays (2d)



- A two-dimensional array may be stored in one-dimensional computer memory in either of two ways:

13

3 dimensional



14

Why do we use arrays?

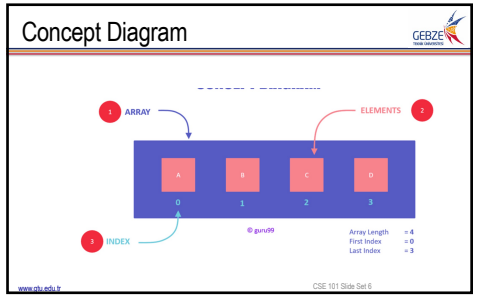
- Arrays are used **when there is a need to use many variables of the same type**.
- It can be defined as a sequence of objects which are of the same data type.
- It is used to store a collection of data, and it is more useful to think of an array as a collection of variables of the same type

15

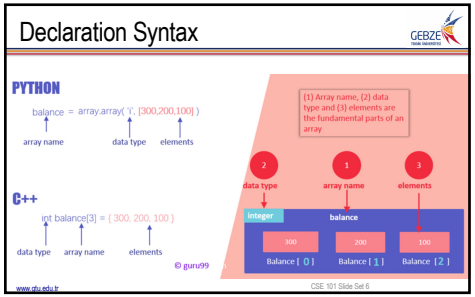
Use of Arrays

- An array is a data structure for storing more than one data item that has a similar data type.
- The items of an array are allocated at adjacent memory locations.
- These memory locations are called **elements** of that array.
- The total number of elements in an array is called **length**.
- The details of an array are accessed about its position.
- This reference is called **index** or **subscript**.

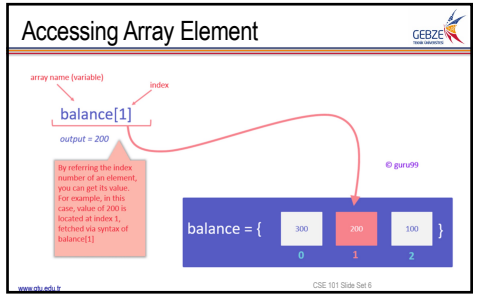
16


Concept Diagram
GEBZE
1 ARRAY
ELEMENTS 2
A
B
C
D
0
1
2
3
3 INDEX
© guru99
Array Length = 4
First Index = 0
Last Index = 3
www.gtu.edu.tr
CSE 101 Slide Set 6

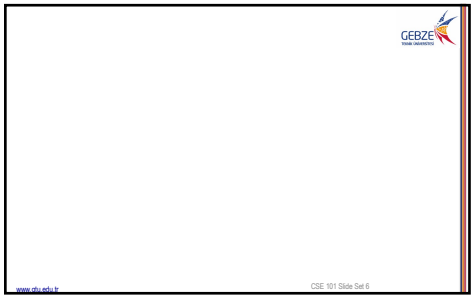
17


Declaration Syntax
GEBZE
PYTHON
balance = array.array('i', [300,200,100])
array name
data type
elements
(1) Array name, (2) data type and (3) elements are the fundamental parts of an array
2 data type
1 array name
3 elements
C++
int balance[3] = { 300, 200, 100 }
data type
array name
elements
© guru99
integer
balance
300
200
100
Balance [0]
Balance [1]
Balance [2]
www.gtu.edu.tr
CSE 101 Slide Set 6

18


Accessing Array Element
GEBZE
array name (variable)
index
balance[1]
output = 200
By referring the index number of an element, you can get its value. For example, in this case, value of 200 is located at index 1, fetched via syntax of balance[1]
© guru99
balance = {
300
200
100
}
0
1
2
www.gtu.edu.tr
CSE 101 Slide Set 6

19


GEBZE
www.gtu.edu.tr
CSE 101 Slide Set 6

20

Question

- Write a program to count the number of occurrences of each digit, of white space characters (blank, tab, newline), and of all other characters.

21

```
#include <stdio.h>

/* count digits, white space, others */
main()
{
    int c, i, nwhite, nother;
    int ndigit[10];

    nwhite = nother = 0;
    for (i = 0; i < 10; ++i)
        ndigit[i] = 0;

    while ((c = getchar()) != EOF)
        if (c >= '0' && c <= '9')
            ++ndigit[c-'0'];
        else if (c == ' ' || c == '\n' || c == '\t')
            ++nwhite;
        else
            ++nother;

    printf("digits =");
    for (i = 0; i < 10; ++i)
        printf(" %d", ndigit[i]);
    printf(", white space = %d, other = %d\n",
        nwhite, nother);
}
```


22

Output

```
digits = 9 3 0 0 0 0 0 0 0 1, white space = 123, other = 345
```


23

24

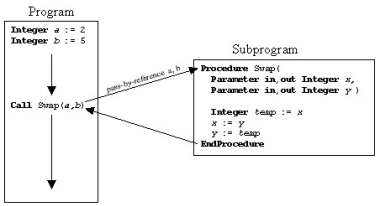
Functions

```c
#include<stdio.h>

int multiply(int a, int b);

int main()
{
    ... ...
    result = multiply(i, j);
    ... ...
}

int multiply(int a, int b)
{
    ... ...
    return a*b;
}
```

The value returned by the function must be stored in a variable.

25

Why function? And how its done in machine lang



26

Machine code function call/jsr/bsr

```
0 movv sp 80
1 movv a 100
2 movv b 400
3 call 20
4 add a b
5 out acc
6 halt


20 push a
21 push b
22 push acc
23 movv a 20
24 movv b 80
25 add a b
26 out acc
27 pop acc
28 pop b
29 pop a
30 ret
```

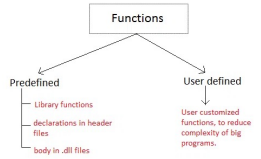

27

Functions

- A function provides a convenient way to encapsulate some computation, which can then be used without worrying about its implementation.
- With properly designed functions, it is possible to ignore how a job is done; knowing what is done is sufficient.
- C makes the sue of functions easy, convinient and efficient; you will often see a short function defined and called only once, just because it clarifies some piece of code.

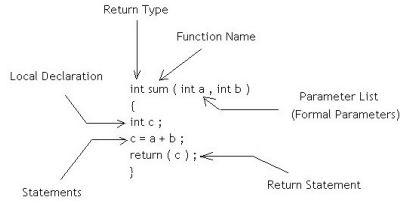
28

Types of Functions



29

Details of Functions



30

Function Examples and Q&A

31

32

33

34

35

36

37

38