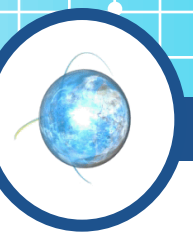
yantox_ska@yahoo.com

Array

Session 9

Arrays

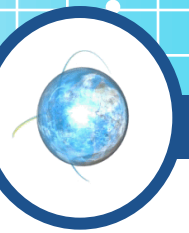
yantox_ska@yahoo.com



- Array index starts with zero
- The last index in an array is num – 1 where num is the no of elements in a array
- int a[5] is an array that stores 5 integers
- a[0] is the first element where as a[4] is the fifth element
- We can also have arrays with more than one dimension
- float a[5][5] is a two dimensional array. It can store 5x5 = 25 floating point numbers
- The bounds are a[0][0] to a[4][4]

Structures

yantox_ska@yahoo.com



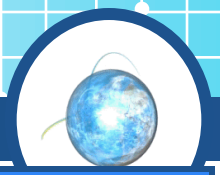
- Structures are user defined data types
- It is a collection of heterogeneous data
- It can have integer, float, double or character data in it
- We can also have array of structures

```
struct <<structname>>
{
    members;
}element;
```

We can access element.members;

Arrays

yantox_ska@yahoo.com



Arrays in C are composed of a particular type, laid out in memory in a repeating pattern. Array elements are accessed by stepping forward in memory from the base of the array by a multiple of the element size.

```
/* define an array of 10 chars */
char x[5] = {'t','e','s','t','\0'};

/* accessing element 0 */
x[0] = 'T';

/* pointer arithmetic to get elt 3 */
char elt3 = *(x+3);  /* x[3] */

/* x[0] evaluates to the first element;
 * x evaluates to the address of the
 * first element, or &(x[0]) */

/* 0-indexed for loop idiom */
#define COUNT 10
char y[COUNT];
int i;
for (i=0; i<COUNT; i++) {
  /* process y[i] */
  printf("%c\n", y[i]);
}
```

Brackets specify the count of elements. Initial values optionally set in braces.

Arrays in C are 0-indexed (here, 0..9)

x[3] == *(x+3) == 't' (*NOT* 's'!)

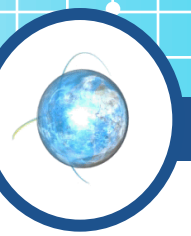
What's the difference between char x[] and char *x?

For loop that iterates from 0 to COUNT-1. Memorize it!

Symbol	Addr	Value
char x [0]	100	't'
char x [1]	101	'e'
char x [2]	102	's'
char x [3]	103	't'
char x [4]	104	'\0'

Introduction to Arrays

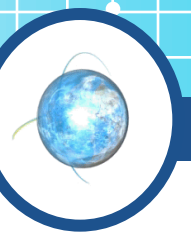
yantox_ska@yahoo.com



- Array definition:
  - A collection of data of same type
- First "aggregate" data type
  - Means "grouping"
  - int, float, double, char are simple data types
- Used for lists of like items
  - Test scores, temperatures, names, etc.
  - Avoids declaring multiple simple variables
  - Can manipulate "list" as one entity

Declaring Arrays

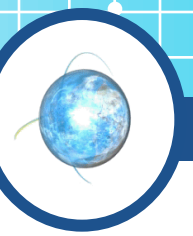
yantox_ska@yahoo.com



- Declare the array → allocates memory
  int score[5];
  - Declares array of 5 integers named "score"
  - Similar to declaring five variables:
    int score[0], score[1], score[2], score[3], score[4]
- Individual parts called many things:
  - Indexed or subscripted variables
  - "Elements" of the array
  - Value in brackets called index or subscript
    - Numbered from 0 to size - 1

Accessing Arrays

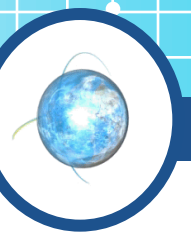
yantox_ska@yahoo.com



- Access using index/subscript
  - printf(“%d”,score[3]);
- Note two uses of brackets:
  - In declaration, specifies SIZE of array
  - Anywhere else, specifies a subscript
- Size, subscript need not be literal
  - int score[MAX_SCORES];
  - score[n+1] = 99;
    - If n is 2, identical to: score[3]

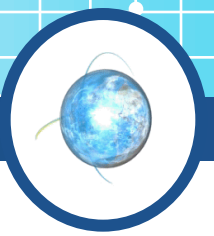
Array Usage

yantox_ska@yahoo.com



- Powerful storage mechanism
- Can issue command like:
  - "Do this to i^{th} indexed variable"
    where i is computed by program
  - "Display all elements of array score"
  - "Fill elements of array score from user input"
  - "Find highest value in array score"
  - "Find lowest value in array score"

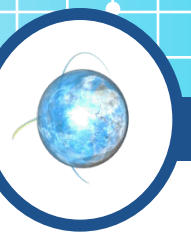
Array Program Example: Display 5.1 Program Using an Array



Display 5.1 Program Using an Array

```
1  //Reads in five scores and shows how much each
2  //score differs from the highest score.
3  #include <iostream>
4  using namespace std;
5  int main( )
6  {
7      int i, score[5], max;
8      cout << "Enter 5 scores:\n";
9      cin >> score[0];
10     max = score[0];
11     for (i = 1; i < 5; i++)
12     {
13         cin >> score[i];
14         if (score[i] > max)
15             max = score[i];
16         //max is the largest of the values score[0],..., score[i].
17     }
```

Array Program Example: Display 5.1 Program Using an Array



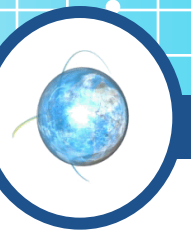
```
18        cout << "The highest score is " << max << endl
19             << "The scores and their\n"
20             << "differences from the highest are:\n";
21        for (i = 0; i < 5; i++)
22            cout << score[i] << " off by "
23                 << (max - score[i]) << endl;
24        return 0;
25    }
```

SAMPLE DIALOGUE

Enter 5 scores:
5 9 2 10 6
The highest score is 10
The scores and their
differences from the highest are:
5 off by 5
9 off by 1
2 off by 8
10 off by 0
6 off by 4

for-loops with Arrays

yantox_ska@yahoo.com



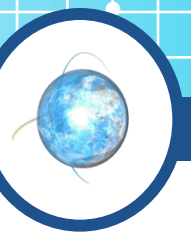
- Natural counting loop
  - Naturally works well "counting thru" elements of an array
- Example:

```
for (idx = 0; idx<5; idx++)
{
      cout << score[idx] << "off by "
                  << max – score[idx] << endl;
}
```

  - Loop control variable (idx) counts from 0 – 5

Major Array Pitfall

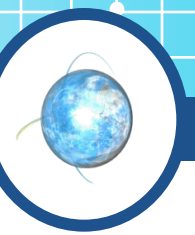
yantox_ska@yahoo.com



- Array indexes always start with zero!
- Zero is "first" number to computer scientists
- C++ will "let" you go beyond range
  - Unpredictable results
  - Compiler will not detect these errors!
- Up to programmer to "stay in range"

Major Array Pitfall Example

yantox_ska@yahoo.com



- Indexes range from 0 to (array_size – 1)
  - Example:
    double temperature[24]; // 24 is array size
    // Declares array of 24 double values called temperature
    - They are indexed as:
      temperature[0], temperature[1] … temperature[23]
  - Common mistake:
    temperature[24] = 5;
    - Index 24 is "out of range"!
    - No warning, possibly disastrous results

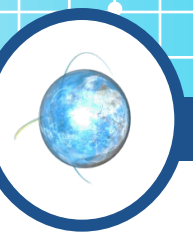
Defined Constant as Array Size

yantox_ska@yahoo.com

- Always use defined/named constant for array size
- Example:
  const int NUMBER_OF_STUDENTS = 5;
  int score[NUMBER_OF_STUDENTS];
- Improves readability
- Improves versatility
- Improves maintainability

Uses of Defined Constant

yantox_ska@yahoo.com



- Use everywhere size of array is needed
  - In for-loop for traversal:

```
for (idx = 0; idx < NUMBER_OF_STUDENTS; idx++)
{
    // Manipulate array
}
```

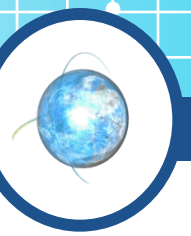
  - In calculations involving size:

```
lastIndex = (NUMBER_OF_STUDENTS – 1);
```

  - When passing array to functions (later)
- If size changes → requires only ONE change in program!

Initializing Arrays

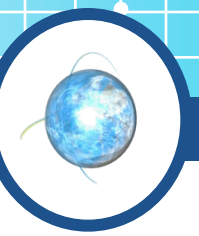
yantox_ska@yahoo.com



- As simple variables can be initialized at declaration:
  int price = 0; // 0 is initial value
- Arrays can as well:
  int children[3] = {2, 12, 1};
  - Equivalent to following:
    int children[3];
    children[0] = 2;
    children[1] = 12;
    children[2] = 1;

Arrays in Functions

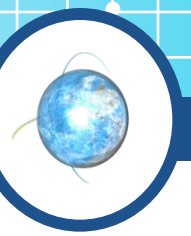
yantox_ska@yahoo.com



- As arguments to functions
  - Indexed variables
    - An individual "element" of an array can be function parameter
  - Entire arrays
    - All array elements can be passed as "one entity"

Indexed Variables as Arguments

yantox_ska@yahoo.com

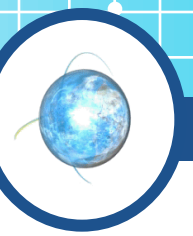


- Indexed variable handled same as simple variable of array base type
- Given this function declaration:
  void myFunction(double par1);
- And these declarations:
  int i; double n, a[10];
- Can make these function calls:

```
myFunction(i);    // i is converted to double
myFunction(a[3]);         // a[3] is double
myFunction(n);   // n is double
```

Entire Arrays as Arguments

yantox_ska@yahoo.com



- Formal parameter can be entire array
  - Argument then passed in function call is array name
  - Called "array parameter"
- Send size of array as well
  - Typically done as second parameter
  - Simple int type formal parameter

Entire Array as Argument Example:

Display 5.3 **Function with an Array Parameter**

SAMPLE DIALOGUEFUNCTION DECLARATION

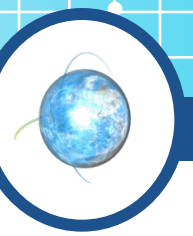
```
void fillUp(int a[], int size);
//Precondition: size is the declared size of the array a.
//The user will type in size integers.
//Postcondition: The array a is filled with size integers
//from the keyboard.
```

SAMPLE DIALOGUEFUNCTION DEFINITION

```
void fillUp(int a[], int size)
{
    cout << "Enter " << size << " numbers:\n";
    for (int i = 0; i < size; i++)
        cin >> a[i];
    cout << "The last array index used is " << (size - 1) << endl;
}
```

Entire Array as Argument Example

yantox_ska@yahoo.com



- Given previous example:
- In some main() function definition, consider this calls:

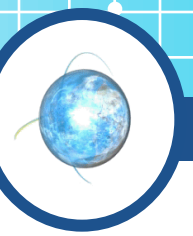
```
int score[5], numberOfScores = 5;
fillup(score, numberOfScores);
```

  - 1st argument is entire array
  - 2nd argument is integer value

- Note no brackets in array argument!

Array as Argument: How?

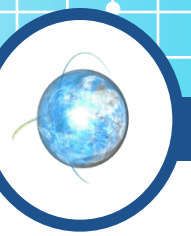
yantox_ska@yahoo.com



- What's really passed?
- Think of array as 3 "pieces"
  - Address of first indexed variable (arrName[0])
  - Array base type
  - Size of array
- Only 1st piece is passed!
  - Just the beginning address of array
  - Very similar to "pass-by-reference"

Array Parameters

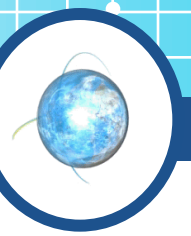
yantox_ska@yahoo.com



- May seem strange
  - No brackets in array argument
  - Must send size separately
- One nice property:
  - Can use SAME function to fill any size array!
  - Exemplifies "re-use" properties of functions
  - Example:
    int score[5], time[10];
    fillUp(score, 5);
    fillUp(time, 10);

Summary

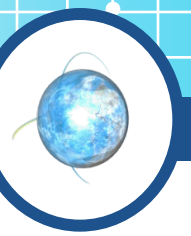
yantox_ska@yahoo.com



- Array is collection of "same type" data
- Indexed variables of array used just like any other simple variables
- for-loop "natural" way to traverse arrays
- Programmer responsible for staying "in bounds" of array
- Array parameter is "new" kind
  - Similar to call-by-reference

Exercise 8.1

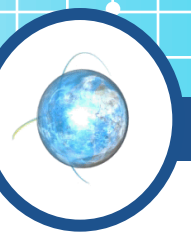
yantox_ska@yahoo.com



```
1.  /*Materi  : Type Data Array
2.    Kasus   : Menentukan bilangan terbesar dari bilangan-bilangan yang
3.              tersimpan dalam variabel array.
4.  */
5.
6.  #include<stdio.h>
7.  #include<conio.h>
8.
9.  void main()
10. {
11.    int bil[15] ;
12.    int x,Max ;
13.    clrscr();
```

Exercise 8.2

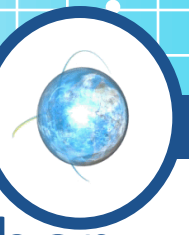
yantox_ska@yahoo.com



```
14. //Memberi nilai variabel array
15.     bil[1]=17; bil[2]=23; bil[3]=20;bil[4]=10;bil[5]=15;
16.     bil[6]=5;  bil[7]=33; bil[8]=95;bil[9]=8;bil[10]=88;
17.     // Menampilkan Bilangan dari var Array
18.     for(x=1;x<=10;x++)     {
19.                         printf("Bilangan Array ke %2d adalah : %2d\n",x,bil[x]);
20.     }
21.     //Menentukan Bilangan Terbesar
22.     Max=0;
23.     for(x=1;x<=10;x++)     {
24.                         if(bil[x] > Max) Max = bil[x];
25.     }
26.     printf("\nBilangan Terbesarnya adalah : %d",Max);
27.     getch();
28. }
```

Task 8

yantox_ska@yahoo.com



- Write a program to find the minimum number from n numbers that entries from keyboard.