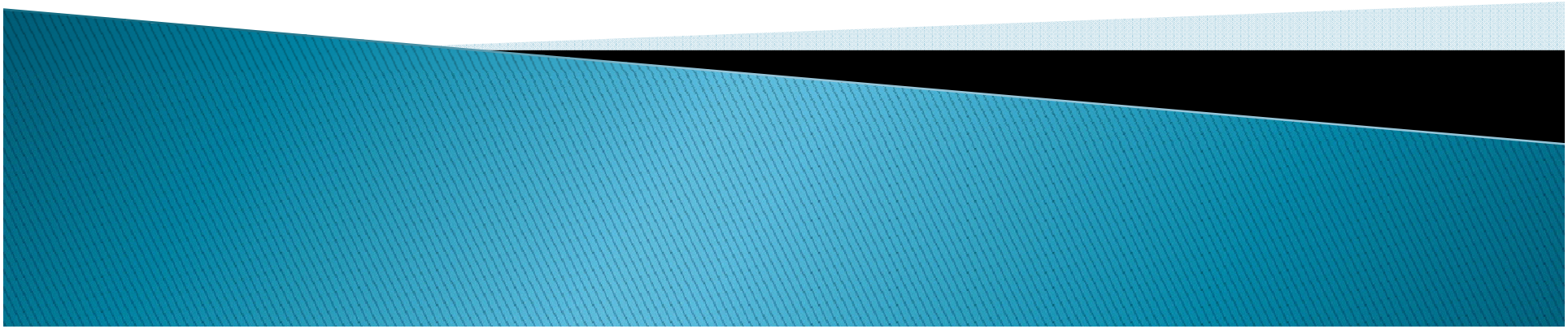
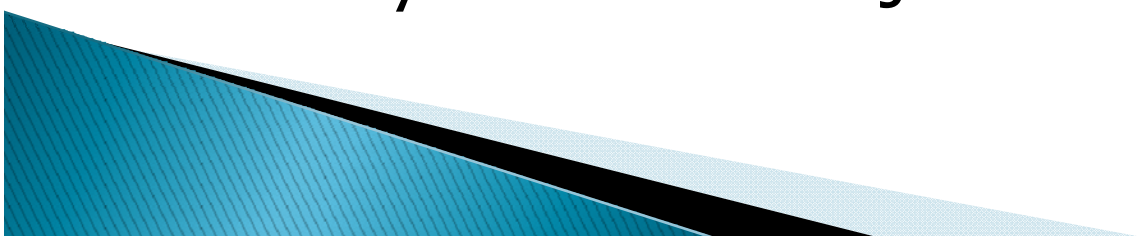
Computer Science 210

C++ part 2



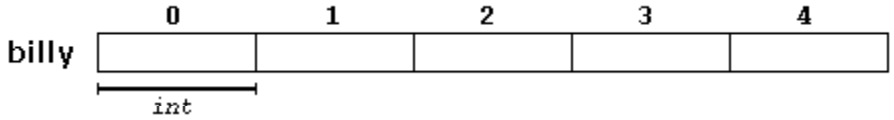
Operators

- Assignment (=)
  - a=5;
- Arithmetic operators (+, -, *, /, %)
- Compound assignment (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)
- Increase and decrease (++, --)
- Relational and equality operators (==, !=, >, <, >=, <=)
- Logical operators (!, &&, ||)
- Conditional operator (?)
- Mostly the same as Java.



Array

- An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.
- type name [elements]; →int billy [5];



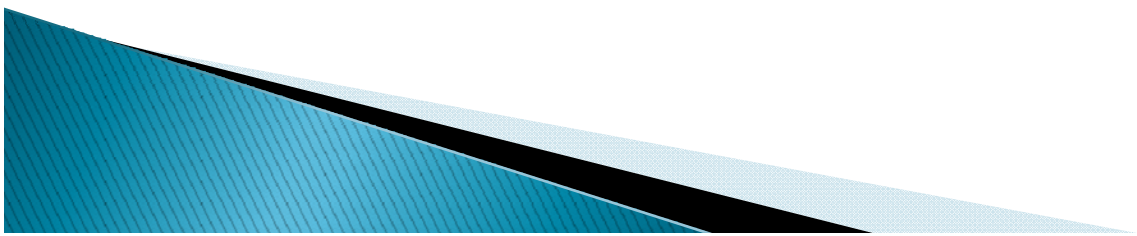
- where each blank panel represents an element of the array, that in this case are integer values of type int. These elements are numbered from 0 to 4 since in arrays the first index is always 0, independently of its length.

String as an array of characters

- char question[] = "Please, enter your first name: ";
- char greeting[] = "Hello, ";
- char yourname [80];
- cout << question;
- cin >> yourname;
- cout << greeting << yourname << "!";
  - question[0] = 'P', question[1] = 'l', question[2] = 'e'…

```
Please, enter your first name: John
Hello, John!
```

- The + operator does concatenation.
  - string str1 = "qwe";
  - string str2 = "rty";
  - string str3;
  - str3 = str1 + str2; // str3 = "qwerty"



Macro

- Simply replacing piece of code
- Implements with cares
- Start with
- #define NAME(x) [code]
- Check **macro.cpp**

Side Effect

Square Macro (without the inner set of parentheses)

- SQR(x) (x * x)
- Parameters: (3+4, 3+4)
- Result = 19 WRONG ANSWER. WHY?

```
#define SQR(x)  (x * x)
...
result = SQR(3+4);
printf("%d\n", result);
```

```
#define SQR(x)  (x * x)
...
result = (3 + 4 * 3 + 4);

printf("%d\n", result);
```

Must use parentheses

```
#define SQR(x)  ((x) * (x))
...
result = ((3 + 4) * (3 + 4));
printf("%d\n", result);
```

Side Effect (con't)

Square Macro (without the outer set of parentheses)

- ADD_FIVE(a) (a) + 5
- Parameters: 3
- Result = 18 WRONG ANSWER. WHY?

```
#define ADD_FIVE(a)  (a) + 5
...
result = ADD_FIVE(3) * 3;
```

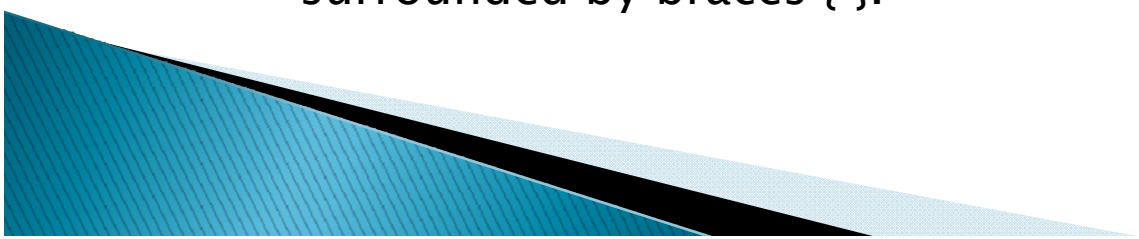
```
#define ADD_FIVE(a)  (a) + 5
...
result = 3)+ 5 * 3;
```

Must use parentheses

```
#define ADD_FIVE(a)  ((a) + 5)
...
result = ADD_FIVE(3) * 3;
```

Functions

- A function is a group of statements that is executed when it is called from some point of the program.
- The following is its format:
  - type name (parameter1, parameter2, ...) { statements }
  - where:
    - type is the data type specifier of the data returned by the function.
    - name is the identifier by which it will be possible to call the function.
    - parameters (as many as needed): Each parameter consists of a data type specifier followed by an identifier, like any regular variable declaration (for example: int x) and which acts within the function as a regular local variable. They allow to pass arguments to the function when it is called. The different parameters are separated by commas.
    - statements is the function's body. It is a block of statements surrounded by braces { }.

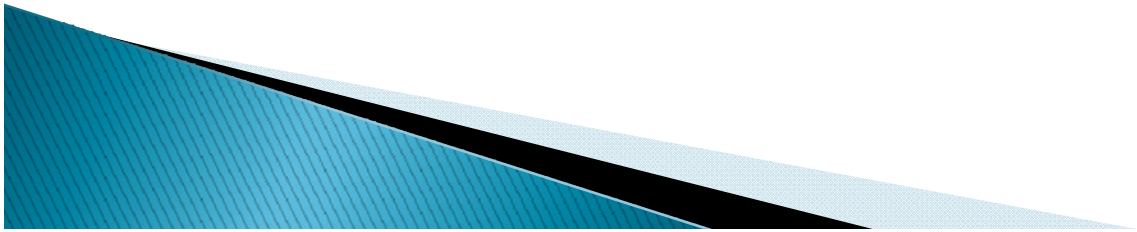


Functions (1)

```
// function example
#include <iostream>
using namespace std;

int addition (int a, int b)
{
  int r;
  r=a+b;
  return (r);
}

int main ()
{
  int z;
  z = addition (5,3);
  cout << "The result is " << z;
  return 0;
}
```

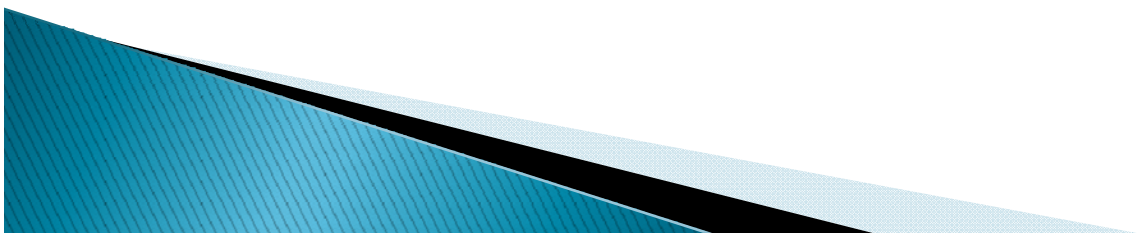
```
// function example
#include <iostream>
using namespace std;

int addition (int a, int b);
int main ()
{
  int z;
  z = addition (5,3);
  cout << "The result is " << z;
  return 0;
}
int addition (int a, int b)
{
  int r;
  r=a+b;
  return (r);
}
```

- Function must be implemented before calling or
- Signature of the function is declared before calling.



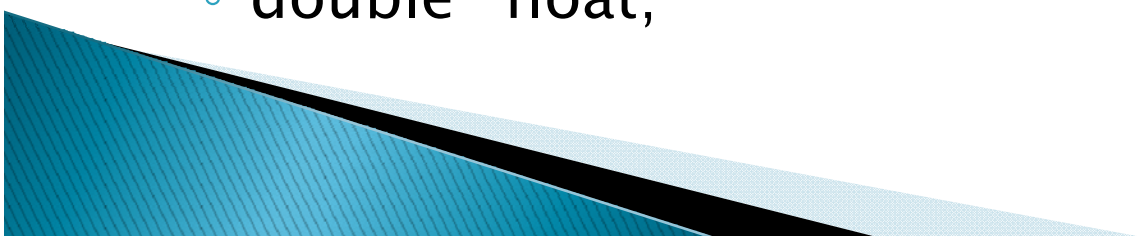
Pointers

- Pointers
  - Pointer are a fundamental part of C/C++.
  - If you cannot use pointers properly then you have basically lost all the power and flexibility that C/C++ allows. The secret to C/C++ is in its use of pointers.
- C/C++ uses pointers a lot. Why?:
  - It is the only way to express some computations.
  - It produces compact and efficient code.
  - It provides a very powerful tool.
- C/C++ uses pointers explicitly with:
  - Arrays,
  - Structures,
  - Functions.
- *NOTE: Pointers are perhaps the most difficult part of C/C++ to understand. C/C++'s implementation is slightly different DIFFERENT from other languages.*



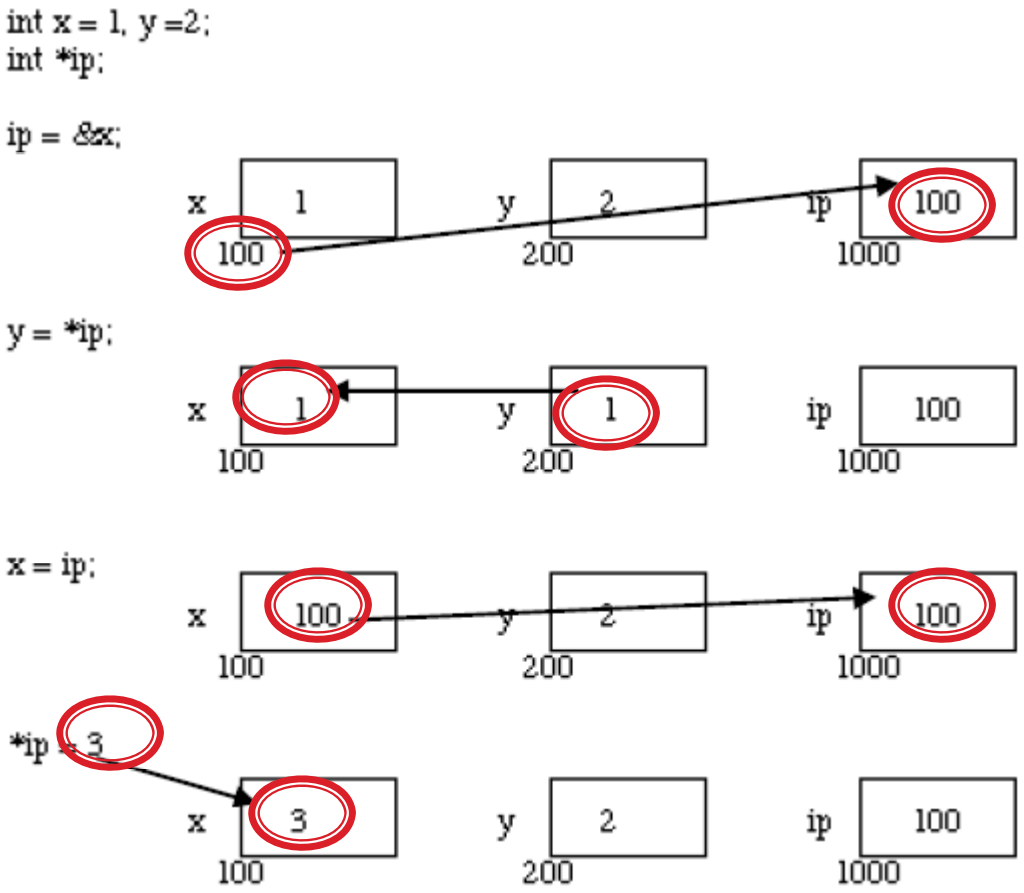
Pointer (2)

- A pointer is a variable which contains the address in memory of another variable.
- We can have a pointer to any variable type.
  - The unary or monadic operator & gives the “address of a variable”.
  - The indirection or dereference operator * gives the ``contents of an object pointed to by a pointer''.
- To declare a pointer to a variable do:
  - int *pointer;
  - char *string;
  - double *float;



Example

- int x = 1, y = 2;
- int *ip;
- ip = &x;
- y = *ip;
- x = ip;
- *ip = 3;
- Check pointer.cpp



Explanation

- Pointer, Variables and Memory Now the assignments x = 1 and y = 2 obviously load these values into the variables. ip is declared to be a pointer to an integer and is assigned to the address of x (&x). So ip gets loaded with the value 100.
- Next y gets assigned to the contents of ip. In this example ip currently points to memory location 200 (the address location of x). So y gets assigned to the values of x (which is 1).
- We have already seen that C is not too fussy about assigning values of different type. Thus it is perfectly legal (although not all that common) to assign the current value of ip to x. The value of ip at this instant is 100.
- Finally we can assign a value to the contents of a pointer (*ip).
- More on POINTERS check *http://www.cplusplus.com/doc/tutorial/pointers/*

Inputs from command promp

- It's very popular in Unix
- A quick way to input and output something
- Applied in your assignment

```
sman063@wintermute02% ./a.out 7e
7e:     7e
sman063@wintermute02% ./a.out 89
89:     81 9
sman063@wintermute02% ./a.out 4002
4002:   81 80 2
sman063@wintermute02% ./a.out 7e 7f 80 81 82 3ffe 3fff 4000 4001 4002
7e:     7e
7f:     7f
80:     81 0
81:     81 1
82:     81 2
3ffe:   ff 7e
3fff:   ff 7f
4000:   81 80 0
4001:   81 80 1
4002:   81 80 2
sman063@wintermute02%
```

argc: number of parameters
argv: array of arguments in strings

```
#include <iostream>  // Header for stream I/O.
using namespace std;
//print all arguments
int main (int argc, char *argv[])
{
    for(int i = 1; i < argc; i++){
        cout << 2*atoi(argv[i]) << " ";
        //changes to integer then times 2
    }
    return 0;
}
```

Exercises

- Ex1: Extend last week exercise.
  - Print all prime numbers from 0 to 10000, 100000,1000000?
  - Can you make it to run faster?
- Ex2: Compute roots of quadratic equations:
  - a
  - (

Theorem. *If*

$$ax^2 + bx + c = 0,$$

then

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Exercises

- Start your last assignment, you may use assignment template as a starting point.