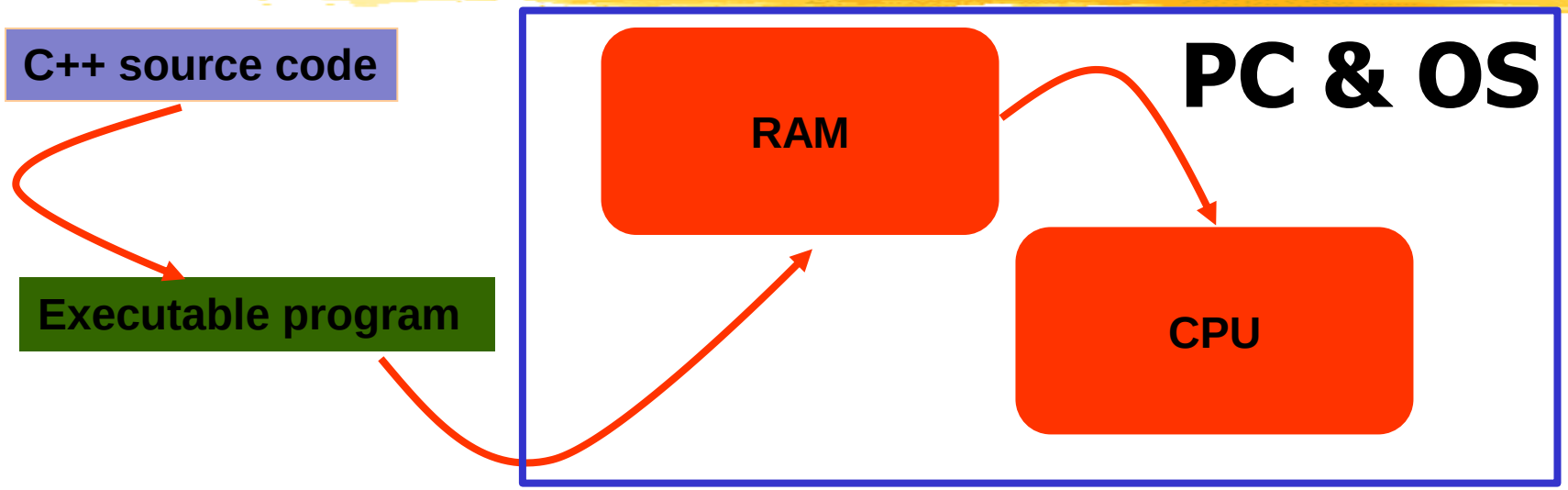
Foundations and advanced C++ programming language

F.S. Cafagna

Intro



- We would like to write, using a given programming grammar, *i.e.* language, a code, in an human readable format …
- … and transform it into something that can be executed by a given processor.
- The latest being named an executable program that can be load by the operating system (**OS**) into a device, *i.e.* **PC** (Personal Computer), memory, *i.e.* **RAM**, and executed by the processor, *i.e.* **CPU**.

Personal Computer

- The Random access memory (RAM) is a group of integrating circuits (board) that implements the storage of data in a random order.
- RAM is volatile. Its content is erased upon PC power down.
- What "random" means? Every data is extracted in a fixed time, independent of memory address or of any relationship with the previously written or read data.

Personal Computer

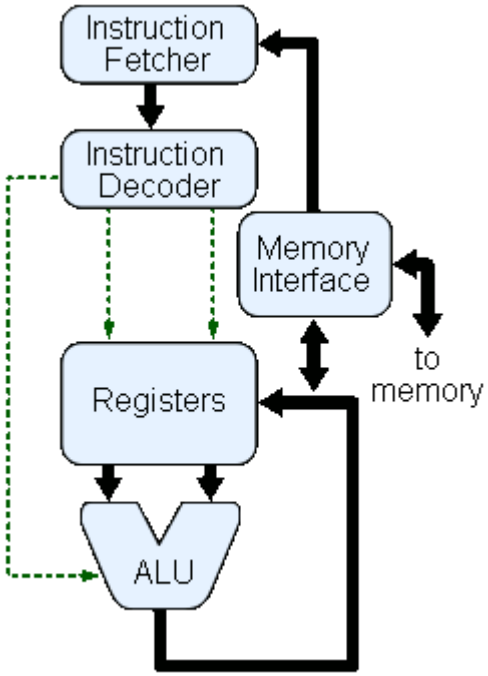
- Data (datum) ?
  - A computer stores any information binary coding it. Any information: text, digit, immagine, audio, etc. etc. it is converted into an ordered (often coded) bit stream or block. The smaller bit block a PC can handle is a byte.
- Bit ?
  - A digit in binary format. It can only be assigned: 0,1
- Byte ?
  - A block of 8 bit. It can span the range from 0 to (2^8-1) (255)
  - $255_{10}=FF_{16}=11111111_2$

Personal Computer

- CPU?
  - Central Processing Unit (CPU).
  - CPU operates on data.
  - It is a logic machine that can execute a finite set of instructions (instruction set architecture - ISA).
  - An ISA is strongly related to programming, and includes the native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O.

Personal Computer

- The possibility of fetch and store instructions and data, increase versatility of a computing unit and lead to a natural need for programming.
- A list of actions, the program code, is translated and coded into a list of low level instructions according to the CPU microcode and registered in memory.



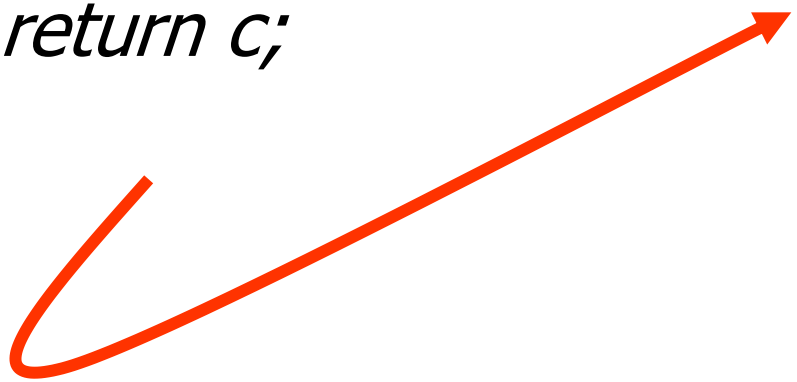
MIPS32 Add Immediate Instruction

001000	00001	00010	0000000101011110
OP Code	Addr 1	Addr 2	Immediate value

Equivalent mnemonic: `addi $r1, $r2, 350`

Personal Comput

```
int main() {
  int a=0,b=2,c=0;
  c= a+b;
  return c;
}
```



```
GAS LISTING /cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s                                                 page 1


 1                              .file   "sum.cpp"
 2                              .stabs  "/home/Cafagna/test/",100,0,0,Ltext0
 3                              .stabs  "sum.cpp",100,0,0,Ltext0
 4                              .text
 5                      Ltext0:
 6                              .stabs  "gcc2_compiled.",60,0,0,0
 7                              .stabs  "__vtbl_ptr_type:t(0,1)=*(0,2)=f(0,3)=r(0,3);-2147483648;2147483647;",128,0,0,0
 8                              .stabs  "bool:t(0,4)=@s8;-16;",128,0,0,0
 9                              .stabs  "wchar_t:t(0,5)=@s16;r(0,5);0;65535;",128,0,0,0
10                              .stabs  "void:t(0,6)=(0,6)",128,0,0,0
11                              .stabs  "long double:t(0,7)=r(0,3);12;0;",128,0,0,0
12                              .stabs  "double:t(0,8)=r(0,3);8;0;",128,0,0,0
13                              .stabs  "float:t(0,9)=r(0,3);4;0;",128,0,0,0
14                              .stabs  "unsigned char:t(0,10)=@s8;r(0,10);0;255;",128,0,0,0
15                              .stabs  "signed char:t(0,11)=@s8;r(0,11);-128;127;",128,0,0,0
16                              .stabs  "short unsigned int:t(0,12)=@s16;r(0,12);0;65535;",128,0,0,0
17                              .stabs  "short int:t(0,13)=@s16;r(0,13);-32768;32767;",128,0,0,0
18                              .stabs  "long long unsigned int:t(0,14)=@s64;r(0,14);0000000000000;01777777777777777777777;",128,0,
19                              .stabs  "long long int:t(0,15)=@s64;r(0,15);0100000000000000000000;0777777777777777777777;",128,0,
20                              .stabs  "long unsigned int:t(0,16)=r(0,16);0000000000000;0037777777777;",128,0,0,0
21                              .stabs  "unsigned int:t(0,17)=r(0,17);0000000000000;0037777777777;",128,0,0,0
22                              .stabs  "long int:t(0,18)=r(0,18);-2147483648;2147483647;",128,0,0,0
23                              .stabs  "int:t(0,3)",128,0,0,0
24                              .stabs  "char:t(0,19)=r(0,19);0;127;",128,0,0,0
25                              .stabs  "__builtin_va_list:t(0,20)=*(0,19)",128,0,0,0
26                              .stabs  "complex long double:t(0,21)=R3;24;0;",128,0,0,0
27                              .stabs  "complex double:t(0,22)=R3;16;0;",128,0,0,0
28                              .stabs  "complex float:t(0,23)=R3;8;0;",128,0,0,0
29                              .stabs  "complex int:t(0,24)=sBreal:(0,3),0,32;imag:(0,3),32,32;;",128,0,0,0
30                              .def    ___main;                        .scl    2;
                                .type   32;                             .endef
31                              .align 2
32                              .stabs  "main:F(0,3)",36,0,1,_main
33                      .globl _main
34                              .def    _main;                          .scl    2;
                                .type   32;                             .endef
35                      _main:
 1:sum.cpp      **** int main() {
36                      .stabn 68,0,1,LM1-_main
37                      LM1:
38 0000 55                             pushl   %ebp
39 0001 89E5                           movl    %esp, %ebp
40 0003 83EC18                         subl    $24, %esp
41 0006 83E4F0                         andl    $-16, %esp
42 0009 B8000000                       movl    $0, %eax
42      00
43 000e 83C00F                         addl    $15, %eax
44 0011 83C00F                         addl    $15, %eax
45 0014 C1EB04                         shrl    $4, %eax
46 0017 C1E004                         sall    $4, %eax
47 001a 8945F0                         movl    %eax, -16(%ebp)
48 001d 8B45F0                         movl    -16(%ebp), %eax
49 0020 E8000000                       call    __alloca
49      00
50                      .stabn 68,0,1,LM2-_main
51                      LM2:
52 0025 E8000000                       call    ___main
52      00
53                      LBB2:
GAS LISTING /cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s                                                 page 2


54                      LBB3:
 2:sum.cpp      ****   int a=0,b=2,c=0;
55                      .stabn 68,0,2,LM3-_main
56                      LM3:
57 002a C745FC00                       movl    $0, -4(%ebp)
57      000000
58 0031 C745F802                       movl    $2, -8(%ebp)
58      000000
59 0038 C745F400                       movl    $0, -12(%ebp)
59      000000
 3:sum.cpp      ****   c=a+b;
60                      .stabn 68,0,3,LM4-_main
61                      LM4:
62 003f 8B45F8                         movl    -8(%ebp), %eax
63 0042 0345FC                         addl    -4(%ebp), %eax
64 0045 8945F4                         movl    %eax, -12(%ebp)
 4:sum.cpp      ****   return c;
65                      .stabn 68,0,4,LM5-_main
66                      LM5:
67 0048 8B45F4                         movl    -12(%ebp), %eax
68                      LBE3:
69                      LBE2:
 5:sum.cpp      **** }
70                      .stabn 68,0,5,LM6-_main
71                      LM6:
72 004b C9                             leave
73 004c C3                             ret
74                      .stabs  "a:(0,3)",128,0,2,-4
75                      .stabs  "b:(0,3)",128,0,2,-8
76                      .stabs  "c:(0,3)",128,0,2,-12
77                      .stabn  192,0,0,LBB3-_main
78                      .stabn  224,0,0,LBE3-_main
79                      Lscope0:
80                      .stabs  "",36,0,0,Lscope0-_main
81                      .text
82                      .stabs "",100,0,0,Letext
83 004d 909090  Letext:
GAS LISTING /cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s                                                 page 3


DEFINED SYMBOLS
                *ABS*:00000000 sum.cpp
/cygdrive/c/DOCUME~1/Cafagna/IMPOST~1/Temp/ccjKWTJ0.s:35     .text:00000000 _main

UNDEFINED SYMBOLS
___main
__alloca
```

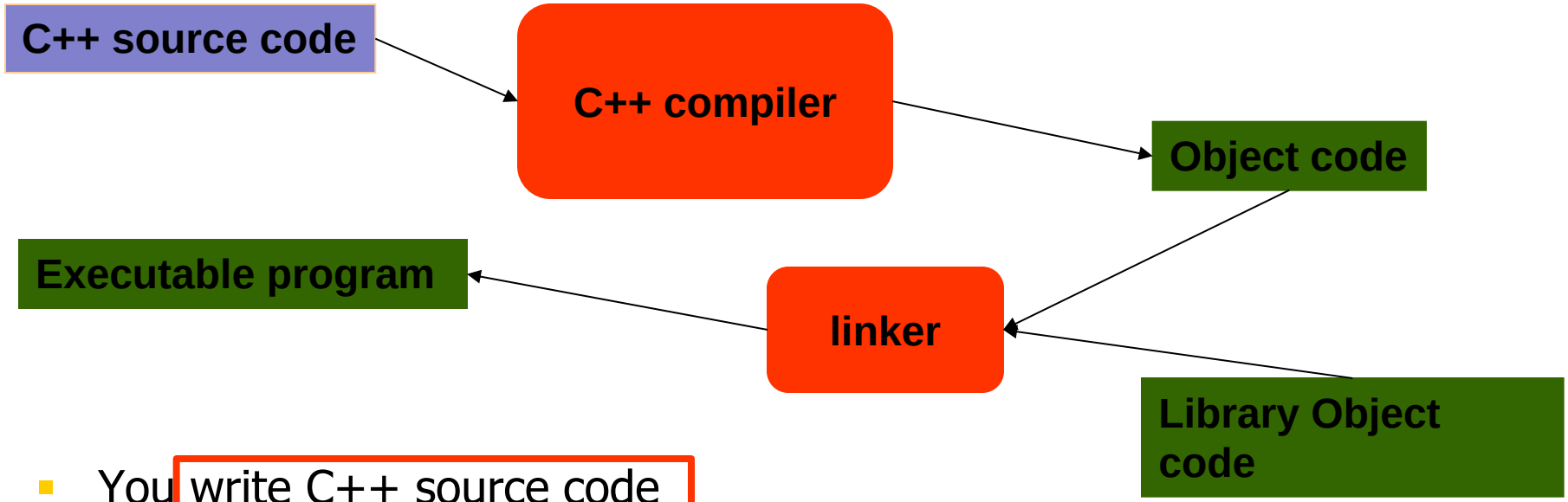
Personal Computer

```
1:sum.cpp       **** int main() {
 36                         .stabn 68,0,1,LM1-_main
 37                         LM1:
 38 0000 55                         pushl       %ebp
 39 0001 89E5                       movl        %esp, %ebp
 40 0003 83EC18                     subl        $24, %esp
 41 0006 83E4F0                     andl        $-16, %esp
 42 0009 B8000000                   movl        $0, %eax
 42      00
 43 000e 83C00F                     addl        $15, %eax
 44 0011 83C00F                     addl        $15, %eax
 45 0014 C1E804                     shrl        $4, %eax
 46 0017 C1E004                     sall        $4, %eax
 47 001a 8945F0                     movl        %eax, -16(%ebp)
 48 001d 8B45F0                     movl        -16(%ebp), %eax
 49 0020 E8000000                   call        __alloca
 49      00
 50                         .stabn 68,0,1,LM2-_main
 51                         LM2:
 52 0025 E8000000                   call        ___main
 52      00
 53                         LBB2:
 54                         LBB3:
  2:sum.cpp      ****    int a=0,b=2,c=0;
 55                         .stabn 68,0,2,LM3-_main
 56                         LM3:
 57 002a C745FC00                   movl        $0, -4(%ebp)
 57      000000
 58 0031 C745F802                   movl        $2, -8(%ebp)
 58      000000
 59 0038 C745F400                   movl        $0, -12(%ebp)
 59      000000
```

Personal Computer

```
3:sum.cpp       ****   c=a+b;
 60                          .stabn 68,0,3,LM4-_main
 61                          LM4:
 62 003f 8B45F8                          movl        -8(%ebp), %eax
 63 0042 0345FC                          addl        -4(%ebp), %eax
 64 0045 8945F4                          movl        %eax, -12(%ebp)
  4:sum.cpp      ****   return c;
 65                          .stabn 68,0,4,LM5-_main
 66                          LM5:
 67 0048 8B45F4                          movl        -12(%ebp), %eax
 68                          LBE3:
 69                          LBE2:
  5:sum.cpp      **** }
 70                          .stabn 68,0,5,LM6-_main
 71                          LM6:
 72 004b C9                              leave
 73 004c C3                              ret
 74                                      .stabs      "a:(0,3)",128,0,2,-4
 75                                      .stabs      "b:(0,3)",128,0,2,-8
 76                                      .stabs      "c:(0,3)",128,0,2,-12
 77                                      .stabn      192,0,0,LBB3-_main
 78                                      .stabn      224,0,0,LBE3-_main
 79                          Lscope0:
 80                                      .stabs      "",36,0,0,Lscope0-_main
 81                                      .text
 82                                      .stabs "",100,0,0,Letext
 83 004d 909090          Letext:
```

Compilation and linking



- You write C++ source code
  - Source code is (in principle) human readable
- The compiler translates what you wrote into object code (sometimes called machine code)
  - Object code is simple enough for a computer to "understand"
- The linker links your code to system code needed to execute
  - E.g. input/output libraries, operating system code, and windowing code
- The result is an executable program
  - E.g. a **.exe** file on windows or an **a.out** file on Unix

So what is programming?

- Conventional definitions
  - Telling a **very** fast moron ***exactly*** what to do
  - A plan for solving a problem on a computer
  - Specifying the order of a program execution
    - But modern programs often involve millions of lines of code
    - And manipulation of data is central
- Definition from another domain (academia)
  - A … program is an organized and directed accumulation of resources to accomplish specific … objectives …
    - Good, but no mention of actually doing anything
- The definition we'll use
  - Specifying the structure and behavior of a program, and testing that the program performs its task correctly and with acceptable performance
    - Never forget to check that "it" works
- Software == one or more programs

Stroustrup/Programming

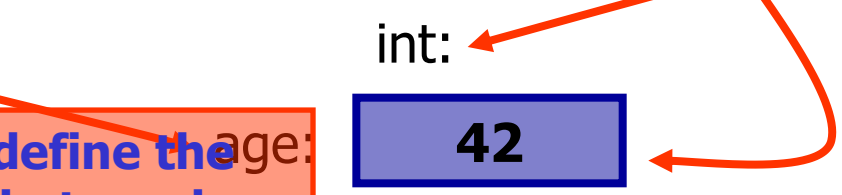
Programming

- Programming, that is, the ideals, techniques, and tools of expressing ideas in code.
- Programming is fundamentally simple
  - Just state what the machine is to do
- So why is programming hard?
  - We want "the machine" to do complex things
    - And computers are nitpicking, unforgiving, dumb beasts
  - The world is more complex than we'd like to believe
    - So we don't always know the implications of what we want
  - "Programming is understanding"
    - When you can program a task, you understand it
    - When you program, you spend significant time trying to understand the task you want to automate
  - Programming is part practical, part theory
    - If you are just practical, you produce non-scalable unmaintainable hacks
    - If you are just theoretical, you produce toys

Stroustrup/Programming

Programming jargon

- To calculate something, we need somewhere to read and write into; i.e. we need a "place" in PC memory to read from or write to. We call such a "place" an *object*.
- An *object* is a region of memory with a *type* that specified what kind of information can be placed in it.
- A named *object* is called a *variable.*
- Think of an object as a "box" into which you can put a value of the object's type:



A type will define the operations that can be executed on that object

Programming jargon

- The most basic building block of a program is an *expression*.
- An *expression* computes a value from a number of operands.
- A part of a code that specifies an action is called a *statement*.

Programming example

- Let's write a program to solve the quadratic equation:

$$ax^2 + bx + c = 0, a \neq 0.$$

- We already know the solutions:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- How many *objects* are present in each of the above equations?
  - 4?
  - 13?
  - 0?
- What is the minimal amount of *expressions*?
  - 1?
  - 10?
  - 14?

Programming example

- We need objects for:
  1. a;
  2. b;
  3. c;
  4. x_1;
  5. $-b$;
  6. $2a$;
  7. ac;
  8. $4ac$;
  9. b^2;
  10. $b^2 - 4ac$;
  11. $\sqrt{b^2 - 4ac}$;
  12. $-b + \sqrt{b^2 - 4ac}$;
  13. $\frac{-b+\sqrt{b^2-4ac}}{2a}$.

- We need expressions for:
  1. Introducing the variable a;
  2. Introducing the variable b;
  3. Introducing the variable c;
  4. Introducing the variable x_1;
  5. $-b$;
  6. $2a$;
  7. ac;
  8. $4ac$;
  9. b^2;
  10. $b^2 - 4ac$;
  11. $\sqrt{b^2 - 4ac}$;
  12. $-b + \sqrt{b^2 - 4ac}$;
  13. $\frac{-b+\sqrt{b^2-4ac}}{2a}$;
  14. $x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}$.

Programming example

- So we can write our first **procedure** (assuming grouping of expression into statements):
  1. Introduce the variable a;
  2. Introduce the variable b;
  3. Introduce the variable c;
  4. Introduce the variable x_1;
  5. Assign a value to a;
  6. Assign a value to b;
  7. Assign a value to c;
  8. Evaluate b^2 and subtract $4ac$;
  9. Evaluate square root of the result of the previous statement;
  10. Evaluate $-b$ and sum it to the result of the previous statement;
  11. Evaluate $2a$ and divide the result of the previous statement by it;
  12. Assign the result of the previous statement to x_1.

Will work?

I need to introduce some checks before using it!

Programming example

So we can write our first procedure (assuming grouping of expression into statements):

1. Introduce the variable a;
2. Introduce the variable b;
3. Introduce the variable c;
4. Introduce the variable x_1;
5. Assign a value to a;
6. Check that a contains a value greater than 0;
7. Assign a value to b;
8. Assign a value to c;
9. Evaluate b^2 and subtract $4ac$;
10. Check that the result of the previous statement is greater than 0;
11. Evaluate square root of the result of the previous statement;
12. Evaluate $-b$ and sum it to the result of the previous statement;
13. Evaluate $2a$, check that the result is greater than 0, and divide the result of the previous statement by it if it is greater than 0;
14. Assign the result of the previous statement to x_1

Will work?

Are we sure a CPU knows how to calculate powers and square roots?

Programming example

So we can write our first procedure:

1. Introduce the variable a;
2. Introduce the variable b;
3. Introduce the variable c;
4. Introduce the variable x_1;
5. Assign a value to a;
6. Check that a contains a value greater than 0;
7. Assign a value to b;
8. Assign a value to c;
9. Evaluate b^2 and subtract $4ac$;
10. Check that the result of the previous statement is greater than 0;
11. Evaluate square root of the result of the previous statement;
12. Evaluate $-b$ and sum it to the result of the previous statement;
13. Evaluate $2a$, check that the result is greater than 0, and divide the result of the previous statement by it if it is greater than 0;
14. Assign the result of the previous statement to x_1.

Will work?

What about digits? Are they variables or something else?

Programming example

So we can write our first procedure:

1. Introduce the variable a;
2. Introduce the variable b;
3. Introduce the variable c;
4. Introduce the variable x_1;
5. Assign a value to a;
6. Check that a contains a value greater than 0;
7. Assign a value to b;
8. Assign a value to c;
9. Evaluate b^2 and subtract $4ac$;
10. Check that the result of the previous statement is greater than 0;
11. Evaluate square root of the result of the previous statement;
12. Evaluate $-b$ and sum it to the result of the previous statement;
13. Evaluate $2a$, check that the result is greater than 0, and divide the result of the previous statement by it if it is greater than 0;
14. Assign the result of the previous statement to x_1.

Will work?

What's about the operation on the objects? Where are results stored?

Programming example

- So we can write our first procedure:

Will work?

What's about the order of the operations?

6. Check that a contains a value greater than 0;
7. Assign a value to b;
8. Assign a value to c;
9. Introduce a temporary variable $t1$;
10. Assign the value contained in b to $t1$;
11. Assign the result of the multiplication of the value contained in b with the one in $t1$ to $t1$;
12. Introduce a temporary variable $t2$;
13. Assign the result of the multiplication of the value contained in a, whit the one contained in c and with 4 to $t2$;
14. Introduce a temporary variable $t3$;
15. Assign the result of the subtraction of the value contained in $t1$ with the one contained in $t2$ to $t3$;
16. Check that $t3$ contains a value greater than 0;
17. Introduce a temporary variable $t4$;
18. Evaluate the square root of the value contained in $t3$ assigning the result to $t4$;
19. Introduce a temporary variable $t5$;
20. Evaluate the sign invertion of the value contained in b assigning the result to $t5$;
21. Assign the result of the sum of the value contained in $t5$ with the one contained in $t4$ assigning the result to $t5$;
22. Introduce a temporary variable $t6$;
23. Assign the result of the multiplication of the value contained in a with 2 assigning the result to $t6$;
24. Check that $t6$ contains a value greater than 0;
25. Assign the result of the division of the value contained in $t5$ with the value contained in $t6$ assigning the result to $x1$;

Programming example

- So we can write our first procedure:

6. Check that a contains a value greater than 0;
7. Assign a value to b;
8. Assign a value to c;
9. Introduce a temporary variable $t1$;
10. Assign the value contained in b to $t1$;
11. Assign the result of the multiplication of the value contained in b with the one in $t1$ to $t1$;
12. Introduce a temporary variable $t2$;
13. Assign the result of the multiplication of the value contained in a, whit the one contained in c and with 4 to $t2$;
14. Introduce a temporary variable $t3$;
15. Assign the result of the subtraction of the value contained in $t1$ with the one contained in $t2$ to $t3$;
16. Check that $t3$ contains a value greater than 0;
17. Introduce a temporary variable $t4$;
18. Evaluate the square root of the value contained in $t3$ assigning the result to $t4$;
19. Introduce a temporary variable $t5$;
20. Evaluate the sign invertion of the value contained in b assigning the result to $t5$;
21. Assign the result of the sum of the value contained in $t5$ with the one contained in $t4$ assigning the result to $t5$;
22. Introduce a temporary variable $t6$;
23. Assign the result of the multiplication of the value contained in a with 2 assigning the result to $t6$;
24. Check that $t6$ contains a value greater than 0;
25. Assign the result of the division of the value contained in $t5$ with the value contained in $t6$ assigning the result to $x1$;

Will work?

What's about the exception handling?

The sqrt?

- How we can calculate the square root of a number?
  - There are several methods. Let's start evaluating roughly a seed approximating the positive real number S we want to calculate the square root.
    - If S ≥ 1, let D be the number of digits to the left of the decimal point;
    - If S < 1, let D be the negative of the number of zeros to the immediate right of the decimal point.
  - Then the rough estimation, being $n = \log_{100} S$, is:
    - D is odd -> D = 2n + 1, then use $\sqrt{S} \approx 2 * 10^n$;
    - D is even -> D = 2n + 2, then use $\sqrt{S} \approx 6 * 10^n$;

    (2 and 6 are used because they approximate the geometric means of the lowest and highest possible values with the given number of digits)
- Heron's method
  - If x is out initial rough guess of $\sqrt{S}$ and e is the error in our estimate than $S = (x + e)^2$ then (assuming e small):

$$e = \frac{S - x^2}{2x + e} \approx \frac{S - x^2}{2x} \rightarrow x \approx x + e = \frac{S + x^2}{2x} = \frac{x + \frac{S}{x}}{2}.$$

  - This became the new guess and we can iteratively update the value until the desired accuracy is obtained (hits is a quadratycaly convergent algorithm).
- Taylor series
- Root-finding algorithm
  - $\sqrt{S}, f(x) = x^2 - S = 0$
- ... etc. etc ...
- Almost all are iterative procedures, i.e. we need iteration statement or the possibility to jump back into our algorithm.

Lessons learned

- A lot is going on in the backstage. A pair of special "magnifier" glasses are needed to unveil all hidden consequences of a design choice, expressions or statements.
- A "standard" set of statements are needed, *i.e.* assignment, conditional statements, iteration statements etc. etc.
- Modularization, *i.e.* factorize common procedures into reusable blocks or libraries.

C++ Historical note and the basic grammar

Historical note

- Who is he?
  - Bjarne Stroustrup, C++ daddy
  - Born in the 1980: "C with Classes", starts to be circulated in the 1983, in the same year was named: C++..
- Since 1990, committees have been created to define C++ standards
  - " … was invented because I wanted to write some event-driven simulations for wich Simula67 would have been ideal, except for efficiency considerations."

Historical note

"C++ was designed primarily so that my friends and I would not have to program in assembler, C, or various modern high-level languages. Its main purpose was to make writing good programs easier and more pleasant for the individual programmer."

B.S., *The C++ Programming language 3rd ed.*

What is C++?

- What is C++ ?
  - C++ is a general-purpose programming language with a bias towards systems programming that:
    - Is a better C;
    - Supports data abstraction;
    - Supports object-oriented programming;
    - Supports generic programming

What is C++?

- What is C++ ?
  - **Procedural Programming**, that is: *Decide which procedures you want; use the best algorithms you can find.*
    - The focus is on the processing, i.e. the algorithm needed to perform the desired computation. A procedural language support this paradigm by providing facilities for passing arguments to functions and returning values from them.
    - C++ improves C, as a procedural language, i.e. I can write a C code.

What is C++?

- What is C++ ?
  - **Modular Programming**, that is: *Decide which modules you want; partition the program so that data is hidden within modules.*
    - Increasing code complexity calls for the need of modularization. Algorithms can be subdivided into blocks implementing part of the procedure hiding data needed in this blocks.
    - Programmer must provide a module interface so the other blocks can use the module, via the interface, or the externally modifiable data, hidden in the module.

What is C++?

- What is C++ ?
  - **Modular Programming**, that is: *Decide which modules you want; partition the program so that data is hidden within modules.*
    - In C++ modules and data can be grouped into namespaces, implementing interfaces and distinguish module of data with identical names but caming from different interfaces.
    - Often modules became so complex that are difficult to maintain. An interface using simply a namespace is not enough.
    - C++ provides facilities to implement a module as an user defined type.

What is C++?

- What is C++ ?
  - **User-Definited Types**, that is: *Decide which types you want; provide a full set of operations for each type.*
    - C++ provides the same support of the base type (integers, floating points, characters etc. etc.) to user defined complex type, so it is possible to use the same rules to manage these types. The most complex of this type is a class
    - Very often modules became user defined types! So that a programmer can store together data and code! Data and procedure implementing calculation on these data. As well as the base type, a module can be dynamically created! When I need the code I can create a memory area to store it along with the data.

What is C++?

- What is C++ ?
  - **User-Definited Types**, that is: *Decide which types you want; provide a full set of operations for each type.*
    - The increase of a type complexity increase the need to safe data and hide procedure details to the user. We want the user focus to be on interfaces, not on the procedure details. Interface must be kept as stable and generic as possible so to avoid changes also if the internal module structure is changing.
    - C++, exploiting the usage of *data abstraction, provides* facilities to create generic interfaces that can have actual different implementations according to the type used.

What is C++?

- What is C++ ?
  - **Object-Oriented Programming**, that is: *Decide which classe you want; provide a full set of operations for each class; make commonality explicit by using inheritance.*
    - If abstract data can be defined also virtual class (abstract) can be. This class do not correspond to any actual class but define an interface that can be used by an user without specification of the actual type.
    - If I have to write code to manage *Inmate*, *Physician*, *Paramedic* and *Employees*, to count the accesses into a building, can I write a code that deals only with *Persons* ?

What is C++?

- What is C++ ?
  - **Generic Programming**, that is: *Decide wich algorithms you want; parametrize them so that they work for a variety of suitable types and data structures.*
    - C++ provides the *template* construct, enabling the possibility of building classes independent from the type they are using.
    - The majority of the C++ libraries: STL (Standard Template Library), exploit this mechanism and provide objects, generic algorithms or facilities that can be specialized by users to a specific type. Because C++ offers to the user defined type the same support of the base type, both can be used.

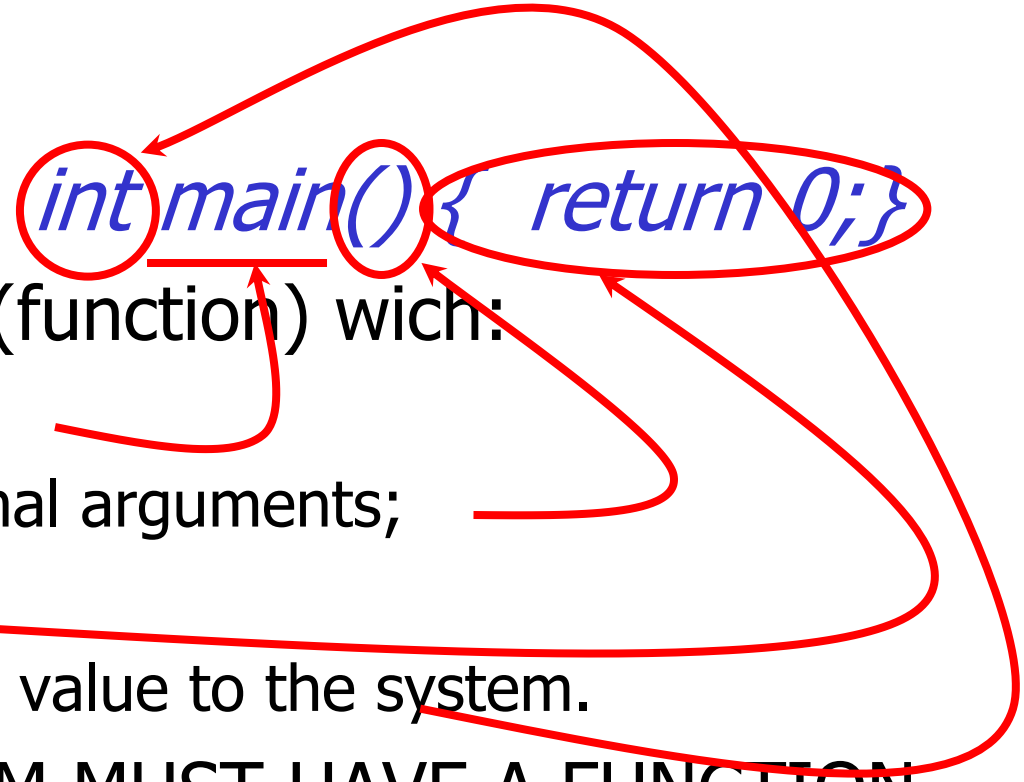
Exercise

- Using the quadratic equation solution write a list of statements that can implement:
  - Procedural programming;
  - Modular programming;
  - Object oriented programming;

Just focus on the expression and statements, write it on simple plain English, but try to implement the above mentioned programming style.

Base grammar

- The minimal code: *int main(){ return 0;}*
  - Defines a module (function) wich:
    - Is called: *main;*
    - Does have no formal arguments;
    - Does nothing;
    - Returns an integer value to the system.
  - ALL C++ PROGRAM MUST HAVE A FUNCTION CALLED: *main()*
  - The executable starts executing such a function.



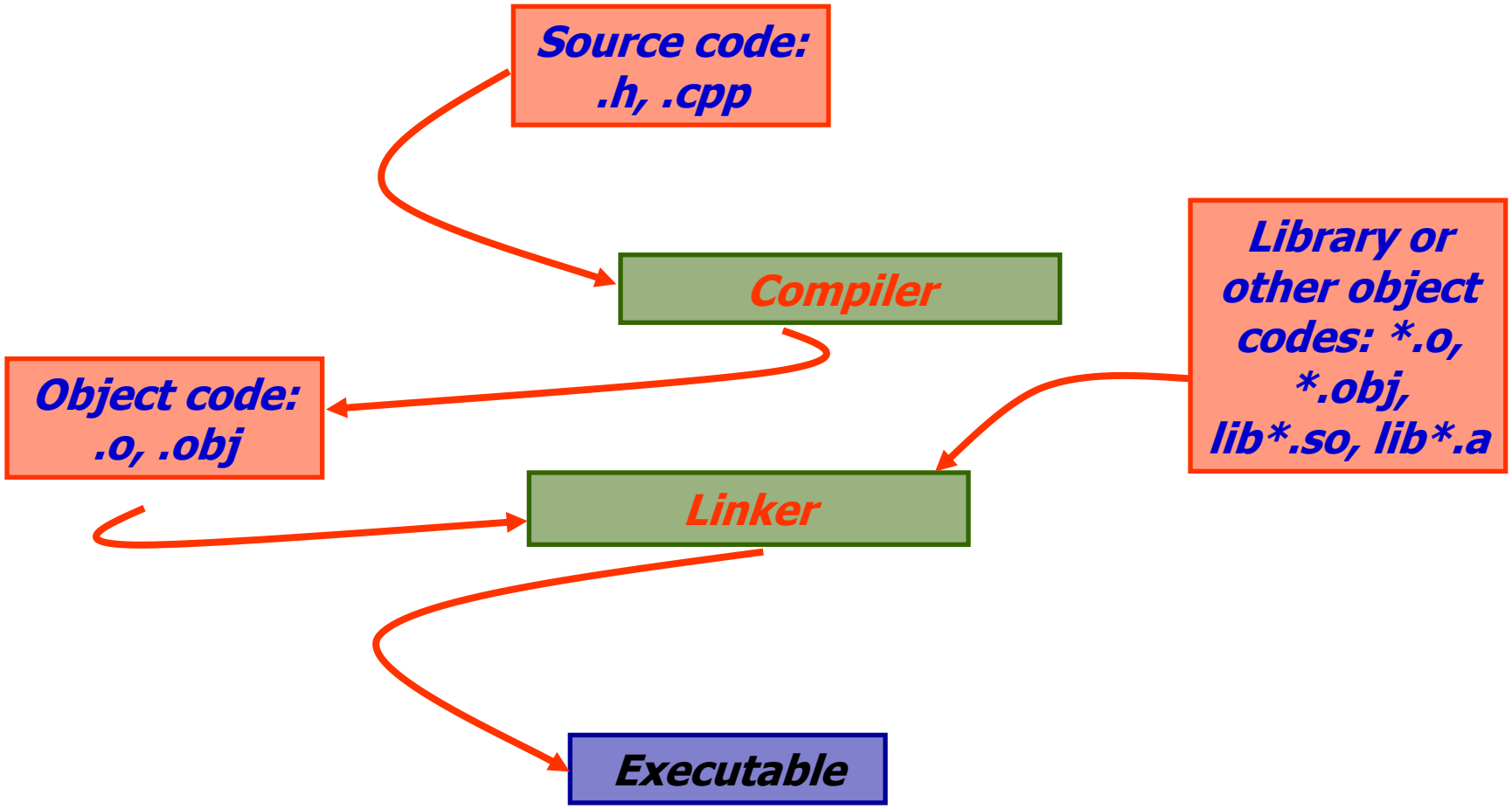
Base grammar

- The minimal code: *int main() { return 0;}*
  - In Unix operating system an executable is considered as successfully executed if return a null value to the system.
  - NOTE: A RETURN VALUE NOT NULL IS CONVENTIONALLY CONSIDERED AS A MARK OF A RUN-TIME FAILURE OF THE MAIN EXECUTION!!!!

Base grammar

- The minimal code: *int main() { return 0; }*
- Curly brackets: *{ }*, they represent a group, the beginning or end of a module.
- They mark a *scope* (**scope** (skōpe), n. **1**. the extent or range of one's understanding. **2**. the area of extent covered by something **3**. opportunity or freedom for movement or activity), the beginning and the end of a module, function, structure, class, etc. etc.

Compilation & Linking

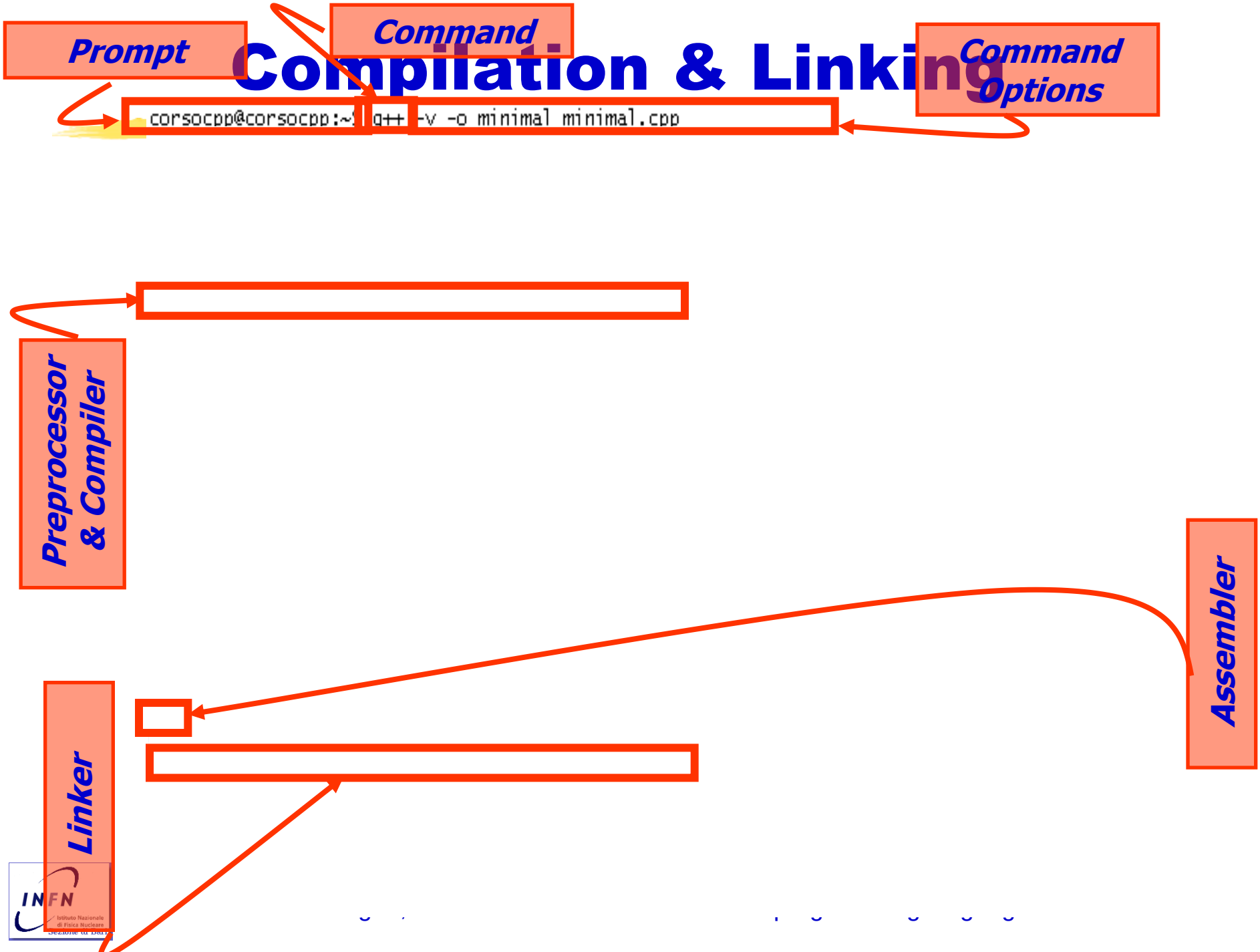


Compilation & Linking

```
corsocpp@corsocpp:~$ g++ -v -o minimal minimal.cpp
Using built-in specs.
Target: i486-pc-linux-gnu
Configured with: ../gcc-4.2.3/configure --prefix=/usr --libexecdir=/usr/lib --infodir=/usr/share/info
--mandir=/usr/share/man --enable-nls --enable-languages=c,c++ --enable-shared --with-system-zlib
--enable-clocale=gnu --enable-objc-gc --enable-__cxa_atexit --enable-threads=posix --with-tune=i486
i486-pc-linux-gnu
Thread model: posix
gcc version 4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/cc1plus -quiet -v -D_GNU_SOURCE minimal.cpp -quiet -dumpbase
minimal.cpp -mtune=i486 -auxbase minimal -version -o /tmp/cc83UfPO.s
ignoring nonexistent directory "/usr/local/include"
ignoring nonexistent directory
"/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../i486-pc-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/i486-pc-linux-gnu
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/backward
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/include
 /usr/include
End of search list.
GNU C++ version 4.2.3 (i486-pc-linux-gnu)
        compiled by GNU C version 4.2.3.
GGC heuristics: --param ggc-min-expand=47 --param ggc-min-heapsize=31860
Compiler executable checksum: 248e0f8ce610fb04dbdddd59d54f3041
 as -V -Qy -o /tmp/ccMHu7OU.o /tmp/cc83UfPO.s
GNU assembler version 2.17.50 (i486-pc-linux-gnu) using BFD version (GNU Binutils) 2.17.50.20070806
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/collect2 --eh-frame-hdr -m elf_i386 -dynamic-linker
/lib/ld-linux.so.2 -o minimal /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crt1.o
/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crti.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/crtbegin.o
-L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3 -L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3
-L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../.. /tmp/ccMHu7OU.o -lstdc++ -lm -lgcc_s -lgcc -lc -lgcc_s
-lgcc /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/crtend.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crtn.o
corsocpp@corsocpp:~$
```

INFN Istituto Nazionale di Fisica Nucleare Sezione di Bari

Compilation & Linking



Compilation & Linking

```
corsocpp@corsocpp:~$ g++ -v -o test_pointer test_pointer.cpp
Using built-in specs.
Target: i486-pc-linux-gnu
Configured with: ../gcc-4.2.3/configure --prefix=/usr --libexecdir=/usr/lib --in
fodir=/usr/share/info --mandir=/usr/share/man --enable-nls --enable-languages=c,
c++ --enable-shared --with-system-zlib --enable-clocale=gnu --enable-objc-gc --e
nable-__cxa_atexit --enable-threads=posix --with-tune=i486 i486-pc-linux-gnu
Thread model: posix
gcc version 4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/cc1plus -quiet -v -D_GNU_SOURCE test_poin
er.cpp -quiet -dumpbase test_pointer.cpp -mtune=i486 -auxbase test_pointer -vers
ion -o /tmp/ccaxZJBe.s
ignoring nonexistent directory "/usr/local/include"
ignoring nonexistent directory "/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../..
/i486-pc-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/i486-pc-linu
x-gnu
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../../include/c++/4.2.3/backward
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/include
 /usr/include
End of search list.
GNU C++ version 4.2.3 (i486-pc-linux-gnu)
        compiled by GNU C version 4.2.3.
GGC heuristics: --param ggc-min-expand=47 --param ggc-min-heapsize=31860
Compiler executable checksum: 248e0f8ce610fb04dbdddd59d54f3041
 as -V -Qy -o /tmp/ccpizVof.o /tmp/ccaxZJBe.s
GNU assembler version 2.17.50 (i486-pc-linux-gnu) using BFD version (GNU Binutil
s) 2.17.50.20070806
 /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/collect2 --eh-frame-hdr -m elf_i386 -dynam
ic-linker /lib/ld-linux.so.2 -o test_pointer /usr/lib/gcc/i486-pc-linux-gnu/4.2.
3/../../../crt1.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../crti.o /usr/lib/
gcc/i486-pc-linux-gnu/4.2.3/crtbegin.o -L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3 -L
/usr/lib/gcc/i486-pc-linux-gnu/4.2.3 -L/usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../.
./.. /tmp/ccpizVof.o -lstdc++ -lm -lgcc_s -lgcc -lc -lgcc_s -lgcc /usr/lib/gcc/i
486-pc-linux-gnu/4.2.3/crtend.o /usr/lib/gcc/i486-pc-linux-gnu/4.2.3/../../../cr
tn.o
corsocpp@corsocpp:~$
```

Output in assembler

Preprocessore & compilatore

Output in object code

Assembler

Linker

Eseguibile

12/06/2017 afternoon