Principles of Compilers Lab

- **Objectives**
  - For helping the process of building the compiler, we will introduce some standard compiler building tools (namely **lex** and **yacc**).
  - These tool are usually available with the C programming language.
  - So the project will be developed using C.
  - A brief introduction to C is also available at the lab web page.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Objectives

- Why we use C?
- Uses of C
- Brief history
- Program evolution

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Objectives – Why use C?

- C has been used successfully for system programming
- Several standard libraries and tools
- Simple syntax
- Code is nearly as fast as coded in assembly languages

"Middle Level" programming

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Objectives – Uses of C

- Operating Systems
- Language Compilers
- Assemblers
- Text Editors
- Print Spoolers
- Network Drivers
- Databases Mangament

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Objectives – Brief History

- C was originally defined for writing the Unix operating system at AT&T Labs, 1972, by Dennis Ritchie and Ken Thompson.
- Based on B and BCPL, implemented for DEC-PDP11
- ANSI C standard in 1983
- In the earlies ´80, it was used as a base for C++.
- In the earlies ´90, it was used as a base for Java.

Objectives – Program Evolution

1. Editing (writing)
2. Compiling
   a) Pre-processing
   b) Compiling
   c) Linking
3. Executing

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure

- Basic components
- Program Layout
- Pre-processor Directives
- Comments
- First Example

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure Basic components

- C only uses the following characters:
  - Letters: A-Z, a-z
  - Numbers: 0-9
  - Basic Operators: + - * / =
  - Pairwise: { } [ ] ( ) < >
  - Space: ` `
  - Others: . , ; : ` $ “ # % & ! _ |

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure Basic components

- **C only uses the following reserved keywords:**

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

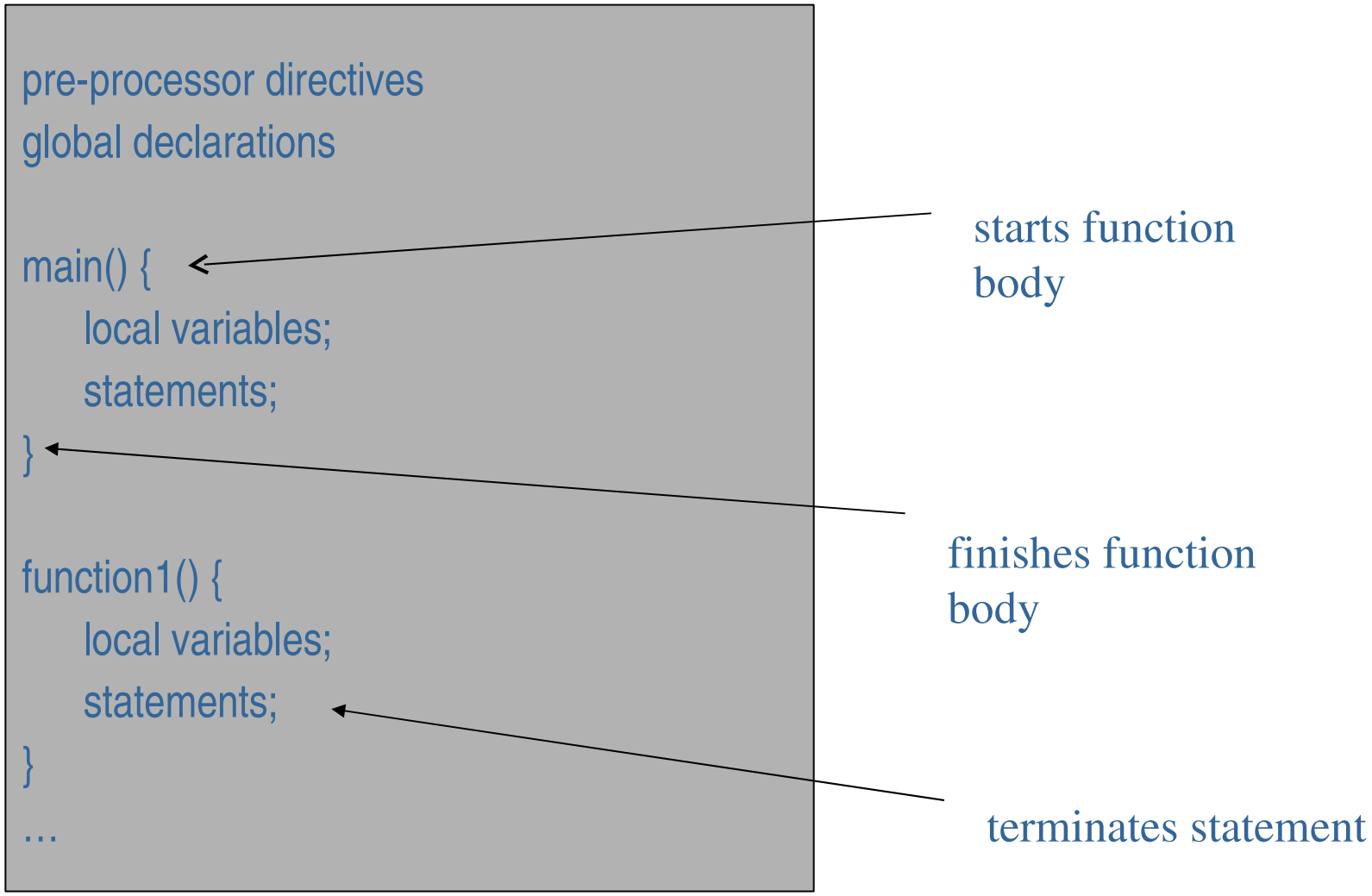
FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure – Program Layout

- **All C program consists of at least one function, called** main().
- main() **is where execution starts.**
- **Other functions may also be in the program.**
- **Each function consists of a set of declarations, and a sequence of statements.**
- **Global declarations (common to all functions) are possible.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure – Program Layout



FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure – Pre-processor Directives

- C is small, several features are not directly included in the language, like text-based input and output.
- C programs are pre-processed before being actually compiled.
- Instructions for this step are called “directives”.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure – Pre-processor Directives

Examples:

- **To include externally defined functions:**

```
#include <stdio.h>
```

- **To define constants:**

```
#define PI 3.141592
```

should be placed in the first column

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure – Comments

- Comments are placed anywhere in a program.
- Must start with /* and end with */, and span through several lines.
- Comments starting with // only last to the end of the current line.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Program Structure – First example

```
/*
   Program hello.c

   A Simple Program
*/

#include <stdio.h>

main() {          // beginning of main

   printf("Hello World!\n");

}                 // end of main
```

FREE UNIVERSITY OF
BOZEN–BOLZANO
Faculty of Computer
Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types

- Built-in Data Types
- Declarations
- Integers
- Floating Point
- Characters
- Assignment
- Expressions
- Example

Primitive Data Types Built-in Data Types

- All variables in C must be declared to be of a type.
- Data types can be built-in, or user-defined.
- Built-in Data Types correspond to elementary information units.
- The particular characteristics of each type depends on the compiler!

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types Built-in Data Types

- **C built-in data types:**
  - int, long, short**: integers**
  - double, float**: floating point numbers**
  - char**: characters**
  - void**: special type with no value**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types Declarations

- **A variable declaration (either local or global) is of the form:**

<type> {<variable_name> [= <initial_value>] }+;

For example:

int a;

float b43;

int pipo=-34;

char a, b,c;

float a=0.01, b=0.02;

Primitive Data Types Integers

- int, long **and** short **values are integers with no fractional part**
- **The range and the memory usage of each type depends on the compiler.**
- **Usually, int ranges from -32768 to 32767.**
- **Operators include +, -, *, /, =, %, ++, -- (prefix and sufix), +=, -=, *=, etc.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types Floating Point

- float, **and** double **are data types for flotaing point number with single and double precision.**
- **The range and the memory usage of each type depends on the compiler.**
- **Usually, doulbe ranges from 1.E-303 to 1.E+303.**
- **Operators include +, -, *, /, =, ++, -- (prefix and sufix), +=, -=, *=, etc**
- **Constants may be written in decimal notacion, or in floating point notation.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types Characters

- char **is the only other built-in data type besides numbers. Strings are managed as arrays of chars.**
- **No special operators are defined (but chars are treated as integers!).**
- **Constants may be written inside single quotes. Example:**

char c = ‘R’;

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types Assignments

- **Assignments change value of variables.**
- **Left hand part contains the variable to change; right hand part contains an expression that when evaluated defines the new value for the variable.**
- **Conversions may take place to make values compatibles.**

```
c = c+10;
p = c/2;
tiempo = 0.001;
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types Expressions

- **An expression is a sequence of operations that returns a value.**
- **Sequences of operations are defined by operator precedence, and parenthesis to change it.**
- **Every variable of a primitive data type can be part of an expression of a primitive data type.**

```
c = 20+s*0.5;
p = 20+(s*0.5);
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Primitive Data Types – Example

```
/*
   Program int.c
   Another simple program using int and printf
*/

#include <stdio.h>

main() {
   int a,b,average;
   a=10;
   b=6;
   average = ( a+b ) / 2 ;
   printf("Here ");
   printf("is ");
   printf("the ");
   printf("answer... ");
   printf("\n");
   printf("%d.",average);
}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output

- **General properties**
- **Function** printf
- **Formatting**
- **Function** scanf
- **Example**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output General properties

- **Standard input/output functions are defined in the library** stdio.h
- **The functions in this libraries are oriented towards a text terminal (for us it will be enough!)**
- **Other libraries are available.**
- **Programmers can define their own i/o functions.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output Function printf

- printf **is used to show some text on the standard output device.**

printf("<string>" { ,<variable>}*)

- **A % character in the string signals that what follows is the value of a variable from the list.**
- **After the %, it is necessary a character specifying how the value should be shown.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output Function printf

- **The list of variables defines the final output of the string. Example:**

```
printf("Hello!");
printf("Total value is %f euro.", total);
printf("Maximun between %d and %d is %d", var1, var2, max);
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output Formatting

- **These are the permitted values after the %:**

%c	char	Single char
%d (%i)	int	Signed int
%e (%E)	float or double	Exponential format
%f	float or double	Signed decimal
%g (%G)	float or double	Use %f or %g as required
%o	int	Unsigned octal value
%x	int	Unsigned hexadecimal
%s	Array of char	Sequence of characters

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output Formatting

- **Before each letter can appear a modifier of the form**

<flag><width>[.<precision>]

- **Flags are – for left justify, + for always display sign, 0 for padding with leading zeros, space for showing a space if there is no sign.**
- **Width is the number of characters used for displaying the value.**
- **Precision for numbers indicates the number of decimal places that will be shown.**
- **Also, in the screen it is possible to include escape characters like \n, \b, \f, \t, \´, \r, etc.**

Input/Output Formatting

- **Examples:**

```
printf("Value: %10.3f", p);
printf("Value: %-10.3f", p);
printf("Value: %+5d", x);
printf("%25s", "Hello");
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output Function scanf

- scanf **is used to read some text from the standard intput device.**

scanf(“<control_string>” { ,<variable>}*)

- **The control string specify how the sequence of characters read from the keyboard should be converted into values.**
- **The rule is that** scanf **processes the input from left to right, and tries to match the characters with a specifier in the control string.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Input/Output Function scanf

- **The width modifier can also be used in the control string.**

```
scanf("%d %d", &i, &j);
scanf("%10d", &i);
```

Input/Output Example

```
#include <stdio.h>

main()  {
   int a,b,c;
   printf("\nThe first number is ");
   scanf("%d",&a);
   printf("\nThe second number is ");
   scanf("%d",&b);
   c=a+b;
   printf("\n The answer is %d \n",c);
 }
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control

- **Logic Expressions**
- **Conditional**
- **Iterations**
- **Sentences** break **and** continue
- **Sentence** switch

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Logic Expressions

- Control statements need to evaluate boolean expressions.
- Boolean expressions are created from basic comparison between variables and/or constants.
- Operators for comparison: ==, >, <, !=, >=, <=
- Operators for boolean composition: !, &, |, &&, ||

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Logic Expressions

- **Since there is no boolean data type, every value different than 0 is considered true.**
- **Examples:**

(a>0)

((b==5) & (c!=1))

((b==5) && (c!=1))

(!done)

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Conditional

- **The conditional sentence has the form:**

if (<condition>) <sentence1>;

[else <sentence2>;]

- **Each sentence may be a compound sentence (sequence of sentences delimited by brackets).**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Conditional

- **Examples:**

```
if (a<b) printf("\n\nFirst number is less than
      second\n\n");
if (a<b) {printf("\n\nFirst number is less than second\n");
           printf("Their difference is : %d\n" , b-a);
           printf("\n");
          };
if (num2 ==0) printf("\n\nCannot devide by zero\n\n");
  else          printf("\n\nAnswer is %d\n\n",num1/num2);
```

Control – Iterations

- **There are three sentences for interation in C:** do, while **and** for**.**

do <sentence>; while (<condition>)

while (<condition>) <sentence>;

for (<expr1>; <expr2>; <expr3>) <sentence>;

- **Each sentence can be a compound sentence, in which case the semicolon is optional.**

Control – Iterations

- **Examples:**

```
#include <stdio.h>

 main()  {
   do
     printf("Hello World!\n");
   while (1 == 1)
  }
```

FREE UNIVERSITY OF
BOZEN–BOLZANO
Faculty of Computer
Science

Control – Iterations

```
#include <stdio.h>

main()   {
  int count;
  count=0;
  while (count < 100) {
     ++count;
     printf("Hello World!\n");
   }
 }
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Iterations

```
#include <stdio.h>

main() {
   int lower , upper , step;
   float fahr , celsius;
   lower = 0  ;
   upper = 300;
   step  = 20 ;

   fahr  = lower;
   while ( fahr <= upper ) {
                        celsius = (5.0 / 9.0) * (fahr - 32.0);
                        printf("%4.0f %6.1f\n" , fahr , celsius);
                        fahr = fahr + step;
                       }
}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Iterations

```
#include <stdio.h>

main()
{
    int fahr;

    for ( fahr = 0 ; fahr <= 300 ; fahr = fahr + 20)
        printf("%4d %6.1f\n" , fahr , (5.0/9.0)*(fahr-32));

}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Iterations

```
#include <stdio.h>

main()  {
  int target;   int guess;   int again;

  printf("\n Do you want to guess a number 1 =Yes, 0=No ");
  scanf("%d",&again);

  while (again)     {
    target = rand() % 100;
    printf('\n What is your guess ? ");
    scanf("%d",&guess);
    while(target!=guess)        {
      if (target>guess) printf("Too low");
      else printf("Too high");
      printf('\n What is your guess ? ");
      scanf("%d",&guess);
     }
    printf("\n Well done you got it! \n");
    printf("\n Do you want to guess a number 1=Yes, 0=No");
    scanf("%d".&again);
   }
 }
```

Control – Sentences break and continue

- **The** break **statement allows to exit a loop from any point of its body, bypassing normal termination.**
- **It´s preferable to use it only in special cases.**
- **The** continue **statement is a kind of opposite of** break. **It forces the next iteration of the loop.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Sentences break and continue

- **Examples:**

```
#include <stdio.h>

main() {
   int t ;

   for ( ; ; ) {
              scanf("%d" , &t)   ;
              if ( t==10 ) break ;
             }
   printf("End of an infinite loop...\n");

}
```

Control – Sentences break and continue

```
#include <stdio.h>

main() {
   int x ;

   for ( x=0 ; x<=100 ; x++) {
                        if (x%2) continue;
                        printf("%d\n" , x);
                       }
}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Sentence switch

- **The** switch **is good for choosing among a set of alternative path which has more than two options.**
- **A variable is sucessively tested against a list of integer or character constants. When a match is found, the statement associated is executed.**

```
switch (<expression>) {
    { case <constant> : <statement_seq>; }*
    [ default : <statement_seq>; ]
}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Control – Sentence switch

- **Example:**

```
#include <stdio.h>

main()  {
  int i;

  printf("Enter a number between 1 and 4");
  scanf("%d",&i);

  switch (i)   {
     case 1:  printf("one");
                    break;
     case 2:  printf("two");
                    break;
     case 3:  printf("three");
                    break;
     case 4:  printf("four");
                    break;
     default: printf("unrecognized number");
   }    // end of switch

}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions

- **Definition**
- **Local variables**
- **Parameters**
- **Sentence** return
- **Invocation**
- **Global variables**
- **Examples**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Definition

- **Functions in C are the only way to structure programs. Any program is a set of functions definition, with a particular initial function (main).**
- **A function is simply a named set of sentences that returns a value, possibly with the definition of local variables.**
- **You can activate the execution of these sentences by this name.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Definition

- **Function communicate with other functions mainly through a set of parameters, and the value returned.**
- **Definition of a function is done with:**

<return_type> <function_name> (<parameter list>)
    <body>

<parameter_list> ::= { <type> <name> },*

<body> ::= [{ { <local_var_definition> |
                    <sentence> };* }]

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Definition

- In C the concept of procedure or subroutine does not exists.
- Functions that are defined with the special return type void are considered as procedures.
- No special consideration is given to these functions.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Local variables

- Local variables are declared inside the body of a function, and can be used only within it.
- Initialization must be done in the declaration, or in an assignment in the body, before its value can be accessed.
- Once the function finishes its execution, the variable exists no longer.
- Each new call to the function creates a set of new local variables.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Parameters

- Inside the body of a function, parameters are treated just as local variables (can be accessed and assigned).
- In the call of a function, the programmer must provide an initial value for these parameters.
- Parameters are passed by copying its value (so modifications inside the function are not seen outside it).

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Sentence return

- **The sentence** return **finishes the execution of a function.**
- **It must be followed by an expression that produces the value returned by the function.**
- **If the function is a procedure, no expression is necessary.**
- **Several of these sentences can be included in the body of a function.**
- **Control is transferred to the calling function, or finishes execution if** return **is in** main**.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Invocation

- **Invocation of a function must be done with:**

<function_name> ({ <expression> },*)

- **If the function returns a non void value, this invocation can be part of an expression.**
- **All function invocations can be considered as an individual sentence (the return value is ignored if it is not a procedure).**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Invocation

- A function must be declared in a program before it is first invoked.
- In some cases this is not possible, so function prototypes (function definitions with empty bodies) are necessary.
- The following definition must match the given prototype.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Global variables

- All variable declaration outside function bodies are considered as global variables for all functions in the program.
- Global variables provides a further communication method between functions (not explicitly defined in function headers).
- Global variables increases the dependences between functions, so should be avoided when possible.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Examples

```
…

int sum(int a, int b)  {
   int result;
   result = a + b;
   return result;
  }

…

var1 = sum(3, c);
sum(c,c);

…
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Functions – Examples

```
…
int sum(int a, int b);
…
var1 = sum(3, c);
sum(c,c);
…

int sum(int a, int b)  {
   int result;
   result = a + b;
   return result;
  }

…
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries

- Input/output
- String manipulation
- Character manipulation
- Advanced Math
- Time and Date
- Miscellaneous functions

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries – Input/output

- **Standard libraries provide function definition for common tasks.**
- **The name of the library must be include in a pre-processor directive.**
- stdio.h **is the library for input/output functions.**
- **Besides** printf **and** scanf **provides the following functions:**
  - getchar()
  - putchar()

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries – String

- string.h **is the library for string manipulation functions.**
- **Strings are array characters.**
- **It provides the following functions:**
  - strcat() // concatenates two strings
  - strcmp() // compares two strings
  - strcpy() // copies one string into another

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries – Character

- ctype.h **is the library for character manipulation functions.**
- **It provides the following functions:**
  - isdigit() // returns a non-zero value if parameter is digit char
  - isalpha()// returns a non-zero value if parameter is letter char
  - isalnum()/* returns a non-zero value if parameter is letter or digit */
  - islower() ()/* returns a non-zero value if parameter is a lowercase letter */
  - isupper() ()/* returns a non-zero value if parameter is an uppercase letter */

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries – Mathematics

- math.h **is the library for advanced mathematical functions.**
- **It provides the following functions:**
  - acos() // returns the arc cosine of parameter
  - asin() // returns the arc sine of parameter
  - atan() // returns the arc tangent of parameter
  - cos() // returns the cosine of parameter
  - exp() // returns the natural logarithm (base e) of parameter
  - fabs() // returns the absolute value of parameter
  - sqrt() // returns the square root of parameter

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries – Time & Date

- time.h **is the library for time and date functions.**
- **It provides the following functions:**
  - time() // returns the current calender time of system
  - difftime()// returns the difference in secs between two times
  - clock() // returns the number of system clock cycles since // program execution

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Standard Libraries – Miscellaneous

- stdlib.h **is the library of miscellaneous functions.**
- **It provides the following functions:**
  - malloc() // provides dynamic memory allocation
  - rand() // provides the next random number in the sequence
  - srand() // used to set the starting point for rand()

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Introduction to C

- Objectives
- Program Structure
- Primitive Data Types
- Input/Output
- Control
- Functions
- Standard Libraries
- Advanced Data Types

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types

- Arrays
- Pointers
- String
- Structures

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Arrays

- **Arrays allow the manipulation of sequences of data items in an uniform way.**
- **Data items in the sequence are accesed by indexes, starting from 0.**
- **An array declaration in C looks like:**

<element_type> <name> [<max_number>];

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Arrays

- Always, the last index of an array is its size minus one (since indexes start at 0).
- Any type (bulit-in or not) can be defined as the element type of an array.
- Character arrays and strings are not the same.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Arrays

- **Arrays are modified and accessed with the following syntax:**

<array_name> [<index>]

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Arrays

```
#include <stdio.h>

main()   {
   int a[5];
   int i;
   for(i =0;i < 5; ++i) scanf("%d",&a[i]);
   for(i =4;i> =0;--i) printf("%d",a[i]);
}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Arrays

```
#include <stdio.h>

main()  {
   char a[5];
   int i;
   for(i=0; i<5; ++i) scanf("%c",&a[i]);
   for(i=4;i>=0;--i) printf("%c",a[i]);
}
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- Pointers are very powerful, but dangerous, data type in C.
- Every variable is seen as an area of memory which has a name.
- So, every variable can be accessed by its name, or the name of the area of memory it is assigned to.

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Declarations:**

int x;

declares a variable of type int with name x

int *y;

declares a variable of type pointer to int with name y.

The same as

int x, *y;

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Access: prefix operators * and &**

x = *y;

assigns the value pointed by y to x

y = &x;

assigns the address of x to y

x = y;

y = x;

should be considered errors

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Example: variable swap (correct?)**

```
…
function swap(int a , int b)  {
  int temp;
  temp = a;
  a    = b;
  b    = temp;
 }
…
int x =10, y=20;
swap(x,y)
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Example: variable swap (correct?)**

```
…
function swap(int *a , int *b)  {
  int temp;
  temp = *a;
  *a   = *b;
  *b   = temp;
 }
…
int x =10, y=20;
swap(x,y)
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Example: variable swap (correct?)**

```
…
function swap(int *a , int *b)  {
  int temp;
  temp = *a;
  *a   = *b;
  *b   = temp;
 }
…
int x =10, y=20;
swap(&x,&y)
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Remember the need to pass to the scanf function the address of the variable to be read, and not ist value**

```
int x;
scanf("%d",&x);
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **There is a close connection between pointers and arrays.**
- **The declaration** int a[10]; **is similar to declaring a pointer to** a[0] **(ie** a **is equal to** &a[0] **).**
- **The only difference is that** a **is a constant pointer; its value cannot be changed.**
- **The expression** a[i] **is converted by the compiler to** *(a+i)**, so it is possible in C to do pointer arithmetic (!?).**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Pointers

- **Example: array of random values**

```
void randdat(int *pa , int n) {
  int i;
  for ( i=0 ; i< n ; ++i)    {
     *pa = rand()%n + 1;
     ++pa;
   }
// or for(i=0; i<n; ++i) *(pa+i)=rand()%n+1;
// or for(i=0; i<n; ++i) pa[i]=rand()%n+1;
}
…
int a[10];
randdat(a);
…
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types String

- There is also a close connection between arrays and strings.
- Strings are considered as character arrays, with a null character (\0) in the last occupied position.
- Strings constant are included in double quotes. (what´s the difference between ´A´ and “A“?)

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types String

- **For copying, comparing, concatenating, and other string manipulation, functions in the standard library** string.h **are provided.** (why a=b, or a==b is not enough?)

- **Example: copying a string constant**

strcopy(a,“hello“);

int a[6]= “hello“;

Advanced Data Types String

- **Example: sorting (I)**

```
#include <stdio.h>

void randdat(int a[] , int n);
void sort(int a[] , int n);
void scan(int a[] , int n , int *done);
void swap(int *a ,int *b);

main()  {
   int i;  int a[20];

   randdat(a , 20);
   sort(a , 20);

   for(i=0;i<20;++i) printf("%d\n" ,a[i]);
 }
...
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types String

- **Example: sorting (II)**

```
...
void randdat(int a[1] , int n)  {
  int i;
  for (i=0 ; i<n ; ++i)
   a[i] = rand()%n+1;
 }

void sort(int a[1] , int n)  {
  int done;
  done = 1;
  while(done == 1) scan(a , n , &done);
 }
...
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types String

- **Example: sorting (III)**

```
…
void scan(int a[1] , int n , int *done)  {
  int i;
  *done=0;
  for(i=0 ; i<n-1 ; ++i)    {
     if(a[i]<a[i+1])       {
       swap(&a[i],&a[i+1]);
       *done=1;
      }
   }
 }
…
```

Advanced Data Types String

- **Example: sorting (IV)**

```
…
void swap(int *a ,int *b)  {
  int temp;
  temp = *a;
  *a   = *b;
  *b   = temp;
 }
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Structures

- **Structures in C allow the creation of data types from heterogeneous elements (records in some other languages).**
- **For defining a new type:**

```
struct <type_name> {
    { <variable_definition> };*  }
```

- **For defining a new variable:**

```
struct <type_name> <variable_name>;
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Structures

- **For accessing a field in a struct variable:**

<variable_name>.<field_name>

- **Struct variables can be assigned with the = operator.**

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Structures

- **Example: complex numbers**

```
…
struct comp {
   float real;    float imag;
 };
…
struct comp add(struct comp a , struct comp b) {
 struct comp c;
 c.real=a.real+b.real;
 c.imag=a.imag+ b.imag;
 return c;
}
…
x = add(y,z);
…
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Structures

- **It is possible to define pointers to structures in the usual way.**
- **Access to structure fields through pointers can be simplify with the -> operator.**

```
struct comp *p;
// then (*p).imag is equivalent to p->imag
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Structures

- **Pointers to structures also allow recursive data types (lists, trees, etc), but then dynamic memory allocation is left to the programmer.**
- **The malloc() function reserves a chunk of memory according to ist parameter, and returns a pointer to it.**

```
int *ptr;
ptr = (*int) malloc(sizeof(int)*10);
```

Advanced Data Types Structures

- **Recursive data structures are useful, but it is cumbersome to manage it in C.**
- **The malloc() function reserves a chunk of memory according to ist parameter, and returns a pointer to it.**

```
int *ptr;
ptr = (*int) malloc(sizeof(int)*10);
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science

Advanced Data Types Structures

- **Example: linked list**

```
…
struct list {
   float data;  struct list *next;
 };
…
struct list *start;
start->data = 3.14;
start->next = (*struct list) malloc(sizeof(list));
start->next->data = 2.71;
start->next->next = (*struct list) malloc(sizeof(list));
…
struct list *cursor; cursor = start;
while (cursor!=0) {
            printf("%10.2f\n", cursor->data);
            cursor = cursor->next;
}
…
```

FREE UNIVERSITY OF BOZEN–BOLZANO
Faculty of Computer Science