ECEA 00002 W/L

“Introduction to Programming”

Structured and Procedural programming with examples in C and C++

Lecturer: dr Marek Piasecki

BASIC CONCEPTS

Program – notation describing the process of converting ***input data*** to ***output data*** according to a certain ***algorithm***.

Input data – Information provided to the program by the user in order to allow the implementation of the algorithm

Output data – are generated by the program, as the results of its execution.

Algorithm – determines how the conversion of input data to output data in accordance with the specified purpose. The algorithm description consists of:

- ***objects*** on which certain actions are performed,
- ***actions*** towards the goal of the algorithm,
- ***flow sequence*** of actions.

Programming – involves coding the ***algorithms*** in the form of ***programs*** understandable for a computer.

Source code – a program written in a specific language such as Java or C, readable for the human / developer

Executable code – program stored as a sequence of commands and data into machine code processor (a computer-readable), usually in the form of binary coded numbers.

The process of creating (coding?) a program:

↓ **editor** → (*.cpp) *source code*
↓ **compiler**→ (*.obj) *object files*
↓ **linker** → (*.exe) *executable code (linked against libraries)*
↓ **debugger** → (step/watch) *tracking the execution, debugging*

C++ is an extension to the **C** language, by adding :

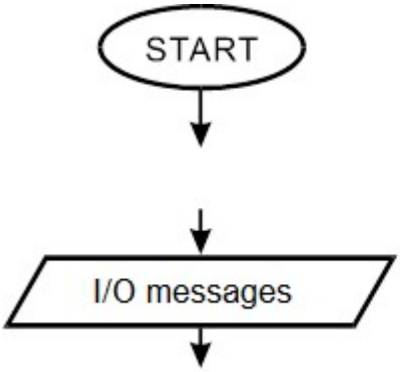
- reference types and variables, anonymous union,
- operators new and delete,
- overloading, functions with the attribute inline,
- default values for funstion arguments,
- function parameters „by reference”,
- classes and objects (**object oriented programming**)
- templates
- error handling > exceptions

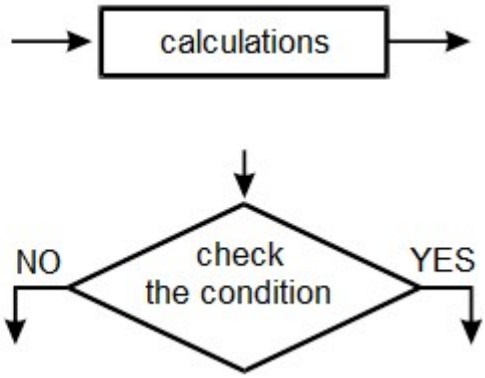
Continuous evolution of programming languages standards: C and C++

YEAR	Language C	Language C++
1972	'improved' B	
1978	Kenigham & Ritchie C	
1979		C with classes
1982→1985		C++
1989	ANSI C	C++ 2.0
1998		C++98
1999	**C99**	
2003		C++03
2007→2011	**C11**	**C++11**
2014		C++14
2017		**C++17**
2018	C18	
2020		C++20
2023	C23	C++23

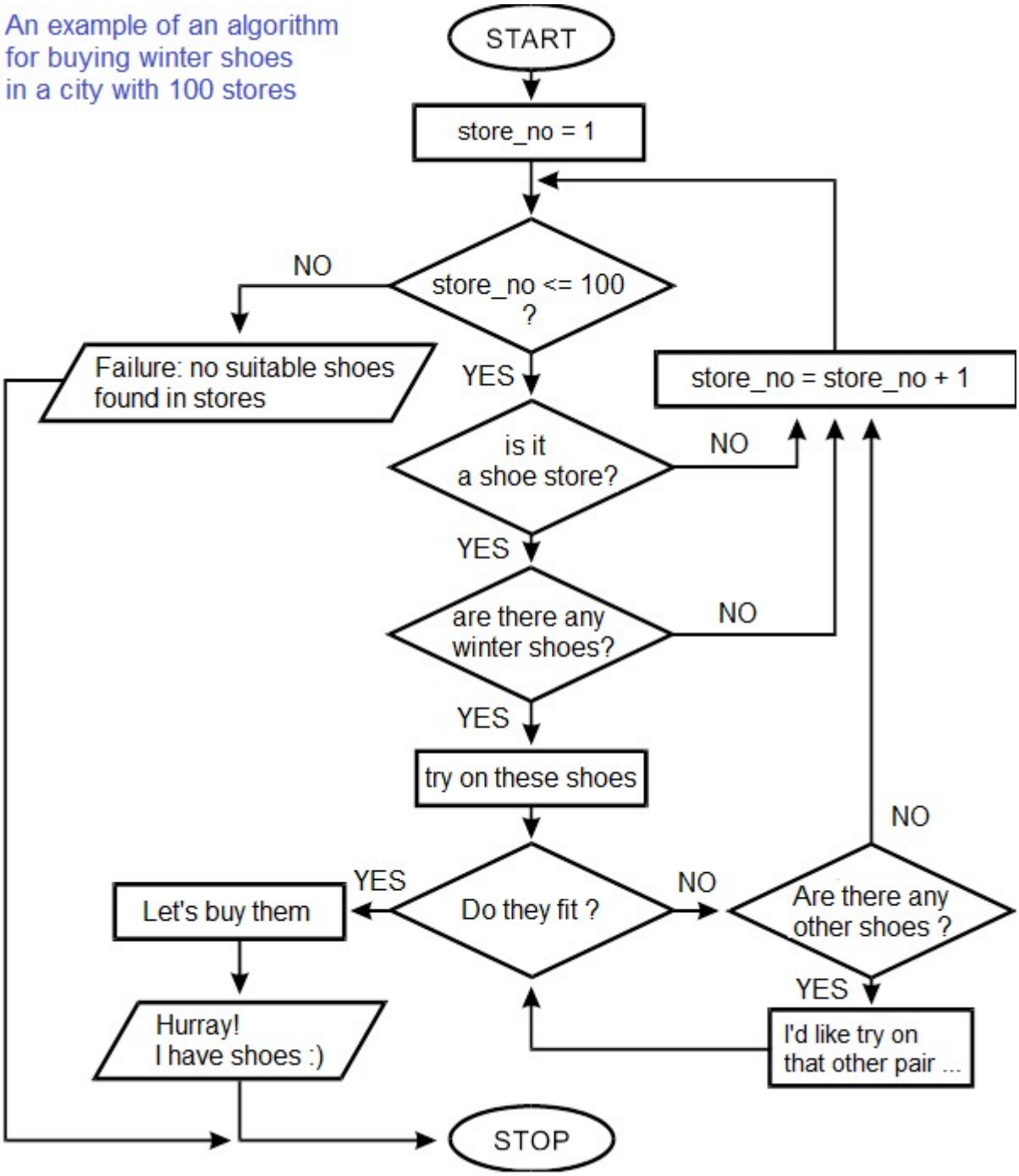
Alternative representation of the algorithm in the form of a flowchart

(*flowchart symbols*)





An example of an algorithm
for buying winter shoes
in a city with 100 stores



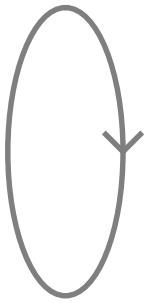
```
int main( ) {  }                                   // the simplest program in C/C++
```

```
int main( int argc, char* argv[ ] )                // with explicit declaration of arguments
{
  return 0;                                        // reporting back, the state of finished program
}
```

```
#include < iostream >                               // C++, print text on the screen
int main()
{
  std::cout << "Hello ! It's me, Your computer" ;   // cout – console output
  std::cin.get();                                  // cin – console input
  return 0;
}
```

```
#include < iostream >          // C++, simple calculation - the product of two numbers
using namespace std;
int main( )
{
  int     number_1, number_2 ;
  float  result ;
  cout << endl << " This is a program calculating product of two numbers " << endl ;
  cout << " Please enter the first number X = " ;
  cin >> number_1 ;
  cout << " Enter a second number       Y = " ;
  cin >> number_2 ;
  result = number_1 * number_2 ;
  cout << endl << " Calculation result  X * Y = " <<  result  << endl ;
  return 0;
}
```

```
#include < iostream >                               // C++, cyclical program execution
using namespace std;
int main( )
{
  char  response;
  do
  {  . . . . . .
       subroutine instructions
     cout << endl << " Do you want to terminate the program (Y / N)? " ;
     cin >> response ;
  } . . . . . .
  while(  response != 'y' ) ;
  cout << endl << " End of the program " ;
  return 0;
}
```



Procedural and Object-Oriented communication with the user

```c
/* procedural:   C / C++  */
#include <stdio.h>
int main(void)
{
  printf("Good ");
  printf("morning!\n");
  getchar();
  return 0;
}
```

```cpp
// object-oriented:   C++
#include <iostream>
int main(void)
{
  std::cout << "Good " ;
  std::cout << "morning" << endl ;
  std::cin.get();
  return 0;
}
```

#include ← directive to paste the text contained in the file

stdio.h ← (**St**andar**dI**nput**O**utput) declaration file of I/O functions

iostream ← (**I**nput**O**utput**Stream**) declaration of O.O. streams

main ← reserved name of the main/initial function

void ← the type of "undefined" or "empty" data

\n ← the transition to a new line

\t ← one "tab" (interspace) is inserted

\" ← quotation mark

**** ← single back slash, e.g. " \ "

endl ← newline "manipulator"

```c
//second example → procedural
#include <stdio.h>

int x,y,s;
int main( )
{
  printf ("Enter  x = ");
  scanf ( "%d" , &x );
  printf ("Enter  y = ");
  scanf ( "%d" , &y );
  s = x+y;
  printf("Result x+y = %d\n", s);

  fflush(stdin);
  getchar();
  return 0;
}
```

```cpp
//second example → OO streams
#include <iostream>
using namespace std;
int x,y,s;
int main( )
{
  cout << "Enter  x = " ;
  cin >> x ;
  cout <<"Enter  y = " ;
  cin >> y ;
  s = x+y;
  cout<< "Result x+y="<< s <<endl;

  cin.ignore( 100, '\n' );
  cin.get();
  return 0;
}
```

Defining variables → determine the name, type, memory reservation

type_name *variableName* **;**
type_name *variable_1, variable_2, variable_3* ;

Basic types: (*for 32-bit applications*)

Type name	Content	Value range	Memory
char	character/letter	-128 ÷ 127	1 byte
int	integer numer	-2147mln ÷ 2147mln	4 bytes
float	real number	10^{-38} ÷ 10^{38} (7 digits)	4 bytes
double	big real number	10^{-308} ÷ 10^{308} (15 digits)	8 bytes

modifiers:

signed	→	with sign (±),	**int**	**char**	–
unsigned	→	without sign,	**int**	**char**	–
short	→	shorter (smaller),	**int**	–	–
long	→	longer (bigger)	**int**	–	**double**

np. **unsigned long int** *largeNumberWithoutSign* ;

Default values:

long	=	**long** int
int	=	signed **int**
char	=	signed **char**

Type	The size in bits	Range
signed char	8	-128 ÷ 127
unsigned char	8	0 ÷ 255
short int	16	-32 768 ÷ 32 767
unsigned short int	16	0 ÷ 65 535
signed int	32	-2 147 483 648 ÷ 2 147 483 647
unsigned int	32	0 ÷ 4 294 967 295
signed long int	32	-2 147 483 648 ÷ 2 147 483 647
unsigned long int	32	0 ÷ 4 294 967 295
float	32	$1.2 * 10^{-38}$ ÷ $3.4 * 10^{+38}$
double	64	$2.2 * 10^{-308}$ ÷ $1.8 * 10^{+308}$
long double	80	3.4 * (10**-4932) ÷ 1.2 * (10**+4932)
bool	8	true, false

OPERATORS

arithmetic operators:

+	addition
–	subtraction
*	multiplication
/	divide numbers
%	the remainder (modulo)

assignment:

=	simple assignment	x = 2;		
+=	assign a sum	x+=2;	→	x = x + 2;
–=	assign a difference	x–=2;	→	x = x – 2;
*=	assign a product	x*=2;	→	x = x * 2;
/=	assign a quotient	x /=2;	→	x = x / 2;
%=	assign a remainder	x%=2;	→	x = x % 2;

Increment and decrement operators:

++variable – increments the variable before calculating the expression

variable++ – increment variable after calculating the expression

––variable – decrement before calculating the expression

variable–– decrement variable after calculating the expression

np. **int** x, y = 1;
x = ++ **y** ; /* *result: x=2, y=2* */ **x = y** ++ ; /* *result: x=1, y=2* */

relational operators:

==	equal
!=	different / not equal
<	smaller then
>	greater than
<=	less than or equal
>=	greater than or equal

logical operators:

&&	conjunction (AND)
||	alternative (OR)
!	negation (NOT)

bitwise logical operators:

&	conjunction (AND)
|	sum (OR)
^	bitwise symmetric difference (XOR)
<<	shift to left (to greater)
>>	shift to right (to smaller)
~	negation of all bits

Priorities of operators in C/C++:

Operator	Description	Example
( )	function call	sin()
[ ]	array element	tab[10]
.	field of structure	person.name
–>	pointing the field of structure	ppinter_to_person–>name
!	logical negation	if(! (x >max)) continue;
~	bitwise negation	~(001101) ≡ (110010)
–	sign change (arithmetic negation)	x = 10 * (– y)
++	increment (increase by 1)	x + + + y ≡ (x+ +) + y
– –	decrement (decrease by 1)	– – y ≠ – – y ≡ – (– y)
&	getting the pointer to variable	ptr_x = &x
*	dereferencing operator	*ptr_x = 10
(type)	type change (*typecast*)	(double) 10 ≡ 10.0
sizeof	the size of variable or type (in bytes)	sizeof(int) ≡ 4
*	multiplication	
/	division	
%	modulo operation (remainder)	if(x%2 == 0) cout << "is even";
+	addition	
–	subtraction	
<<	bitwise left shift	1 << 2 ≡ (0001) << 2 ≡ (0100)
>>	bitwise shift to the right	x = 4 >>1 ≡ x = 2
<	less than	if(number < max) max = number;
<=	less than or equal	
>	greater than	
>=	greater than or equal to	
==	equal	
!=	unequal (different from)	
&	bitwise product	
^	bitwise modulo (exclusive or)	
|	bitwise sum	
&&	logical product	
||	logical sum	
? :	conditional expression	
=	assignment	
*= /= %= += –= <<= >>= &= ^= |=	arithmetic assignments	
,	comma operator	

Example: **int** x=1, y=2, z=3, result=4 ;

result *= –++x*x––+–y––%++z; **(???)**

result *= – (++x) * (x––) +– (y––) % (++z);

result *= (–(++x)) * (x––) + (–(y––)) % (++z);

result *= ((–(++x))*(x––)) + ((–(y––))%(++z)); *// x=1, y=1, z=4, result =–24*