HIGHER SCHOOL OF ECONOMICS

NATIONAL RESEARCH UNIVERSITY

Introduction to Programming

Control-flow Statements

Sergey Shershakov

#4/22 Jan 2019

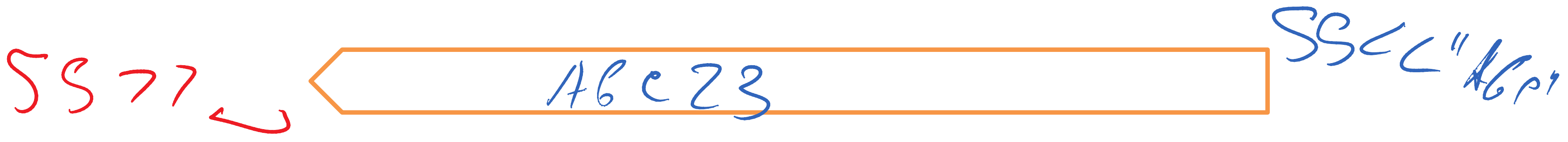
Test 3 (5 pts)

https://goo.gl/forms/9YFM7kohneZGp3Gk2

MORE ON STREAMS AND STRINGS

Class std::stringstream

- Allows one to operate on strings using the general stream approach
- Uses a string buffer that can be read and written
- Thinks of it as StringBuilder in Java and C#



```
#include <sstream>

std::stringstream ss;
ss << "Abc" << 23;          // allows mixing different types together
std::string s = ss.str();   // "Abc23"

std::stringstream ss2;
ss2 << "123";
int n;
ss2 >> n;                   // acts as cin here
```

[http://www.cplusplus.com/reference/sstream/stringstream/]

LOGICAL EXPRESSIONS

Logical Expressions

- A *logical expression* is an expression evaluated as the boolean type
  - contains logical operators;
  - types other than boolean are implicitly converted to `bool`:
    - numbers: 0 → `false`, otherwise `true`
    - pointers: `nullptr` → `false`, otherwise `true`
    - …
- Predicates:
  - (in)equality: `==`, `!=`
  - comparison `<`, `<=`, `>`, `>=`
- Logical Operators:
  - `||` is for logical *OR*
  - `&&` is for logical *AND*
  - `!` is for logical *NOT*

```
%:include <iostream>

int main(int argc, char *argv<::>)
<%
    if (argc > 1 and argv<:1:> not_eq '\0') <%
        std::cout << "Hello " << argv<:1:> << '\n';
    %>
%>
```

The Comparison Operators: == , != , < , <= , > , >=

```
bool l1 = (2 == 2);          ✓
bool l2 = (2. == 2);         ✓
~~bool l3 = ("2" == 2);~~        //
bool l4 = (2. != 22);        ✓
bool l5 = (18 < 42);         ✓

~~bool l6 = ("Abc" < "abc");    //~~

string s1("Abc"), s2("abc");
bool l7 = s1 < s2;    ✓        // strings are compared lexicographically
```

strcmp()

s1.compare(s2)

S2.Compare(S1)

Logical Functions (some examples)

```
bool l10 = isdigit('A');
bool l11 = isdigit('1');
bool l12 = isalpha('A');
bool l13 = isalpha('1');

string s1("Abc");
bool l14 = s1.empty();
```

char ch = 'A';

(int)

The Logical Operators: &&, ||, !

x	y	x && y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x || y
0	0	0
0	1	1
1	0	1
1	1	1

x	!x
~~0~~	
0	1
~~1~~	
1	0

The Logical Operators: &&, ||, !

- The Logical *OR* Operator: ||

```
4 == 4 || 4 == 11 ✓
4 > 3  || 4 > 10 ✓
4 > 9  || 4 < 10 ✓
✓ 4 < 9  || 4 > 2 ✓
~~4 > 9  || 4 < 2~~ ✗
```

- The Logical *AND* Operator: &&

```
6 == 6 && 3 == 3 ✓
6 == 2 && 3 == 3 ✗
6 > 2  && 6 > 10 ✗
6 > 8  && 6 < 10 ✗
6 < 8  && 6 > 2 ✓
6 > 8  && 6 < 2 ✗
```

- Combination of operators:

```
(x > 3 && x < 5) || y > 10

(x > 5 || y > 3) && z > 10

x != 0 && 1.0 / x > 100.0
```

$x || y == !(!x \&\& !y)$

- The Logical *NOT* Operator: !

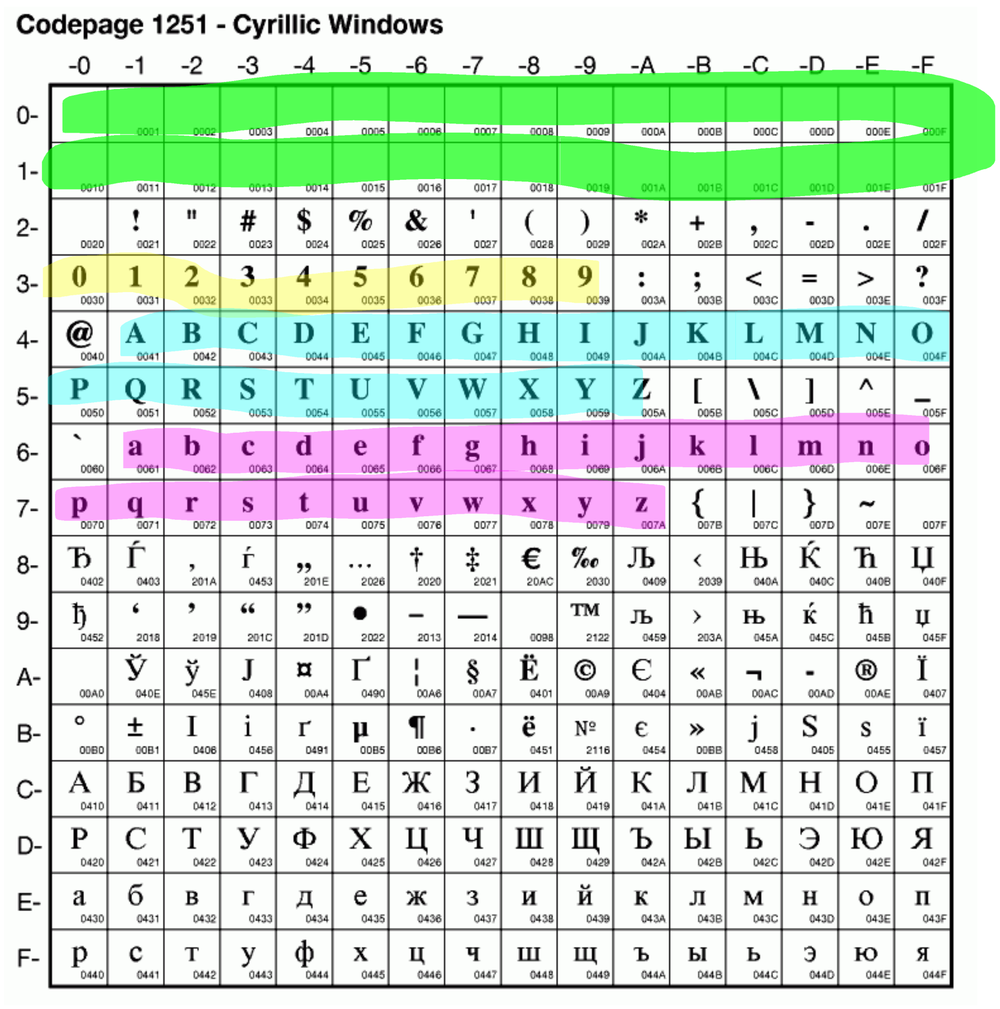
```
!(x > 6) ==> (x <= 6)
!(x > 5)     !x > 5
```

- De Morgan's law:

$\overline{x \vee y} = \bar{x} \cdot \bar{y}$

Setting Up Ranges with Logical Operators

```
char ch = …

if(ch < 32) …

if(ch >= '0' && ch <= '9')…

if(ch >= 'A' && ch <= 'Z'
    ||
   ch >= 'a' && ch <= 'z') …

if(!(ch >= '!' && ch <= '/'))
```



Codepage 1251 - Cyrillic Windows

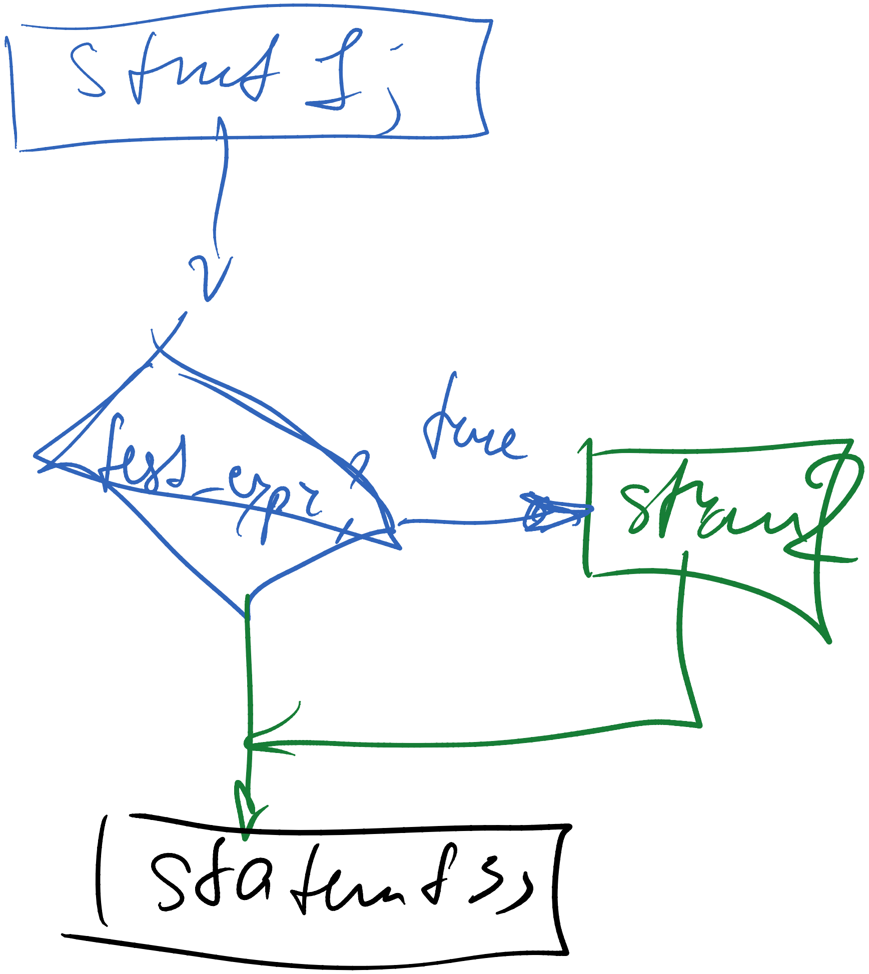
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3-	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
4-	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
5-	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
6-	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
7-	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	007F
8-	Ђ 0402	Ѓ 0403	‚ 201A	ѓ 0453	„ 201E	… 2026	† 2020	‡ 2021	€ 20AC	‰ 2030	Љ 0409	‹ 2039	Њ 040A	Ќ 040C	Ћ 040B	Џ 040F
9-	ђ 0452	‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	0098	™ 2122	љ 0459	› 203A	њ 045A	ќ 045C	ћ 045B	џ 045F
A-	00A0	Ў 040E	ў 045E	Ј 0408	¤ 00A4	Ґ 0490	¦ 00A6	§ 00A7	Ё 0401	© 00A9	Є 0404	« 00AB	¬ 00AC	- 00AD	® 00AE	Ї 0407
B-	° 00B0	± 00B1	І 0406	і 0456	ґ 0491	µ 00B5	¶ 00B6	· 00B7	ё 0451	№ 2116	є 0454	» 00BB	ј 0458	Ѕ 0405	ѕ 0455	ї 0457
C-	А 0410	Б 0411	В 0412	Г 0413	Д 0414	Е 0415	Ж 0416	З 0417	И 0418	Й 0419	К 041A	Л 041B	М 041C	Н 041D	О 041E	П 041F
D-	Р 0420	С 0421	Т 0422	У 0423	Ф 0424	Х 0425	Ц 0426	Ч 0427	Ш 0428	Щ 0429	Ъ 042A	Ы 042B	Ь 042C	Э 042D	Ю 042E	Я 042F
E-	а 0430	б 0431	в 0432	г 0433	д 0434	е 0435	ж 0436	з 0437	и 0438	й 0439	к 043A	л 043B	м 043C	н 043D	о 043E	п 043F
F-	р 0440	с 0441	т 0442	у 0443	ф 0444	х 0445	ц 0446	ч 0447	ш 0448	щ 0449	ъ 044A	ы 044B	ь 044C	э 044D	ю 044E	я 044F

Control-flow Statements

BRANCHING

The if Statement

```
statement1;
if (test_expr)
     statement2;
statement3;
```



- Example:

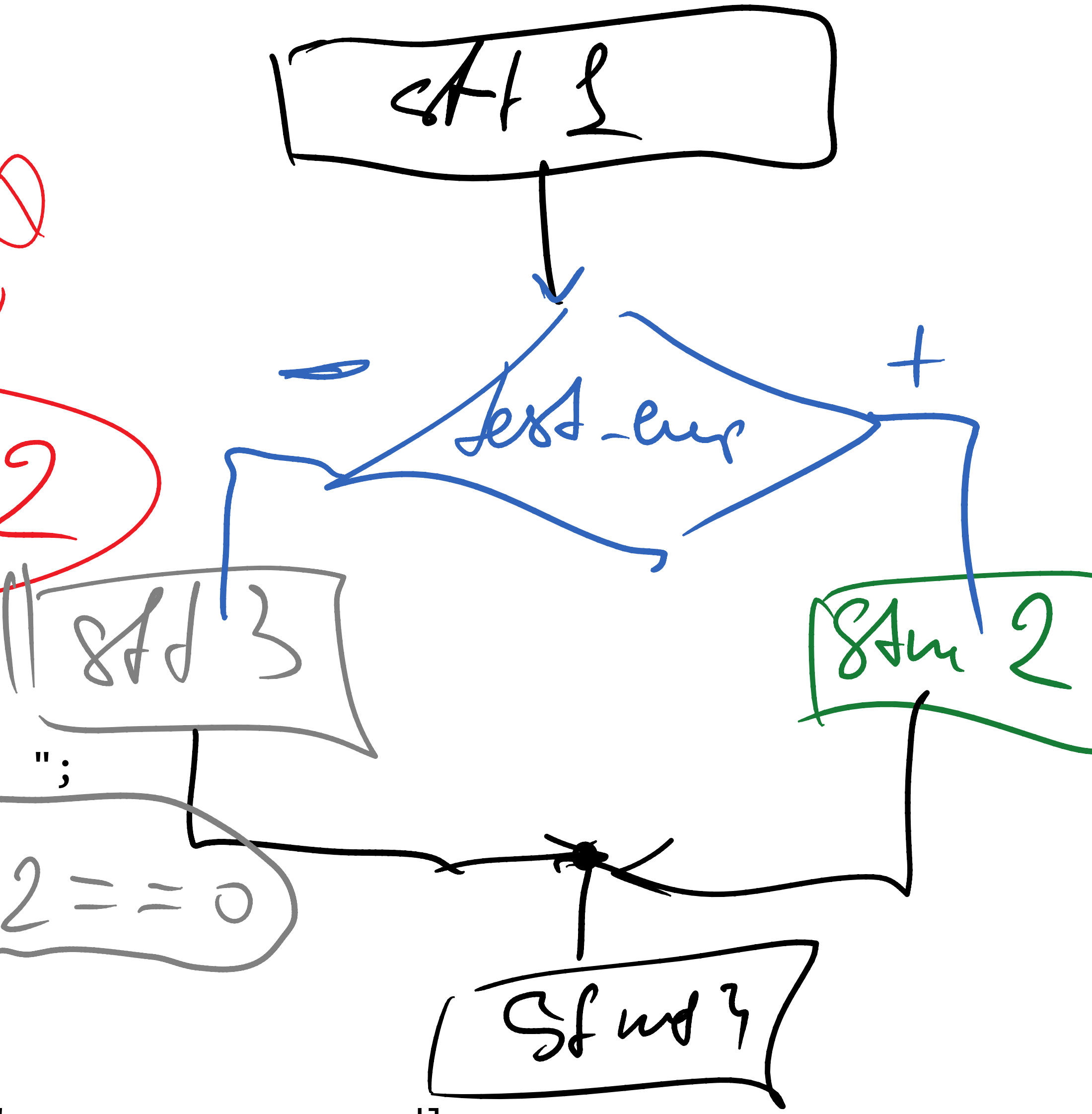
```
int i;
cout << "Input a number: ";
cin >> i;
if(i < 0)
    i = -i;
cout << "The module of the number is: " << i << endl;
```

The `if..else` Statement

```
statement1;
if (test_expr)
      statement2;
else
      statement3;
statement4;
```

0

i % 2



- Example:

```
int i;
cout << "Input a number: ";
cin >> i;
string message;   // i%2==0
if(i % 2)
    message = "odd";
else
    message = "even";
cout << "The number is " << message << endl;
```

Notes on Using `else` Clause

Try to omit using else clause whenever possible!

Good

```
if(x1 == x2 || y1 == y2)
{
    cout << "YES";
}
else if(abs(x) == abs(y))
{
    cout << "YES";
}
else
{
    cout << "NO";
}

return 0;
```

Better!

```
if(x1 == x2 || y1 == y2)
{
    cout << "YES";
    return 0;
}

if(abs(x) == abs(y))
{
    cout << "YES";
    return 0;
}

cout << "NO";

return 0;
```

Notes on Using `else` Clause

Use inversion of logic to keep the conditions of both branches close to each other.

Bad

```
if(x > 0)
{
    cout << "Lorem ipsum dolor sit"
 "amet, consectetur adipiscing"
 "elit, sed do eiusmod tempor"
 "incididunt ut labore et dolore"
 "magna aliqua. Ut enim ad minim"
 "veniam, quis nostrud exercitation"
 "ullamco laboris nisi ut aliquip"
 "ex ea c          onsequat. Duis"

 "in culpa qui officia    erunt"
 "mollit anim id est laborum.";
else
{
    cout << "Hello World";
}
```

5 pages of code!

Better!

```
if(x <= 0)  // !(x > 0)
{
    cout << "Hello World";
else
{
    cout << "Lorem ipsum dolor sit"
 "amet, consectetur adipiscing"
 "elit, sed do eiusmod tempor"
 "incididunt ut labore et dolore"
 "magna aliqua. Ut enim ad minim"
 "veniam, quis nostrud exercitation"
 "ullamco laboris nisi ut aliquip"
 "ex ea commodo consequat. Duis"

 "in culpa qui officia deserunt"
 "mollit anim id est laborum.";
}
```

The ?: Operator

- Serves as a substitution for `if..else` statement when the only need is to have a different expression in one place
  - expression1 ? expression2 : expression3

```
int x;
cout << "Input x = ";
cin >> x;
cout << "Abs(x) is " << (x > 0 ? x : -x);
```

- Is this valid? —

```
cout << " 100 / x = "
     << (x != 0 ? ~~100 / x : "x can't be 0!"~~);
```

The `if..else if..else` Construction

- C++ doesn't have a dedicated elseif clause but it can be implemented by using secondary `if..else` statement as a statement of `else` branch of the first `if..else` statement:

```
if(symb >= '0' && symb <= '9')
    cout << "Digit\n";
else if(symb >= 'A' && symb <= 'Z')
    cout << "Capital Latin Letter\n";
else if(symb >= 'a' && symb <= 'z')
    cout << "Small Latin Letter\n";
else
    cout << "Something else\n";
```

The switch Statement

- The switch statement acts as a routing device switching between different conditions.
- Deals only with integral integer labels;
  - most often, labels are simple char or int constants, or enumerators;
  - the default section (optional) is executed when no other labels don't match the expression;
- Labels can be disjunctively combined:

```
cout << "Press Q to quit: ";
cin >> ch;

switch(ch) {
case 'q':
case 'Q':      ||
     return;
     break;
case …
…
}
```

```
unsigned short day;
cout << "Input day num (1..7): ";
cin >> day;
cout << "Day is ";

switch(day) {
case 1:
     cout << "Monday";     ✓
     ~~break;~~
case 2:
     cout << "Tuesday";   ✓
     ~~break;~~
//... Case 3:     Wed
case 7:
     cout << "Sunday";
     break;
default:
     cout << "Wrong day number";
}
```

How to choose between the `switch` statement and the `if.. else if.. else` construction

```
if(symb >= '0' && symb <= '9')
    cout << "Digit\n";

else if(symb >= 'A' && symb <= 'Z')
    cout << "Capital Latin Letter\n";

else if(symb >= 'a' && symb <= 'z')
    cout << "Small Latin Letter\n";

else
    cout << "Something else\n";
```

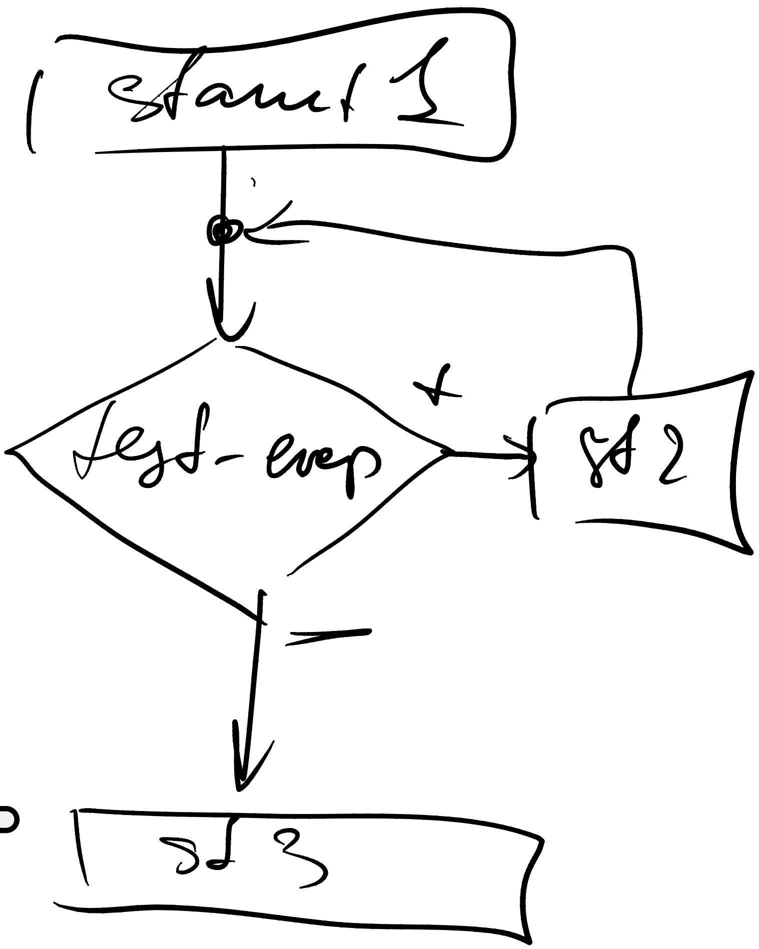
```
unsigned short day;
cout << "Input day num (1..7): ";
cin >> day;
cout << "Day is ";

switch(day) {
case 1:
    cout << "Monday";
    break;
case 2:
    cout << "Tuesday";
    break;
// ...
case 7:
    cout << "Sunday";
    break;
case 8:
case 9:
case 10:
    cout << "Additional Day of a Week :)";
    break;

default:
    cout << "Wrong day number";
}
```

Control-flow Statements

LOOPS

The while Loop

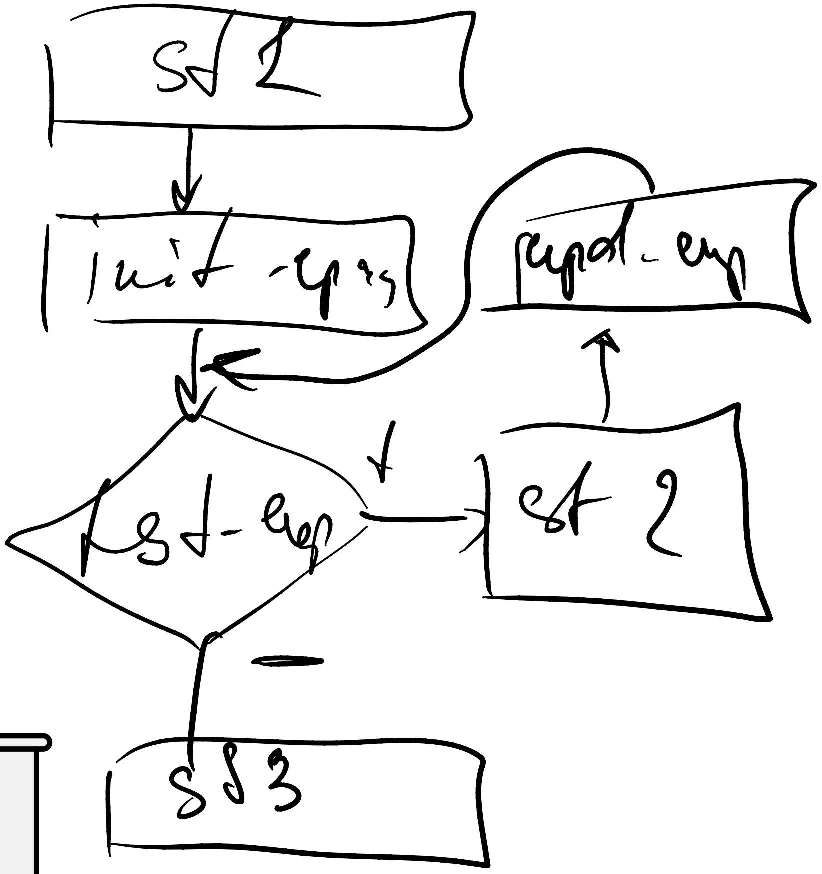
```
statement1;
while (test_expr)
      statement2;
statement3;
```



```
char cstr[] = "Lorem ipsum dolor sit amet...";
int i = 0;
char cur;
while( (cur = cstr[i]) != '\0' )
{
    cout << '\'' << cur << "', ";
    ++i;
}
```

The for Loop

```
statement1;
for (init_expr; test_expr; update_expr)
    statement2;
statement3;
```



```
char cstr[] = "Lorem ipsum dolor sit amet...";

for(int j = 0; j < strlen(cstr); ++j)
    cout << '\'' << cstr[j] << "', ";
```

The for Loop and the while loop

```
statement1;
for (init_expr; test_expr;
          update_expr)
    statement2;
statement3;
```

```
statement1;
init_expr;
while (test_expr)
{
    statement2;
    update_expr;
}
statement3;
```

```
char cstr[] = "Lorem ipsum dolor...";

for(int j = 0; j < strlen(cstr); ++j)
    cout << '\'' << cstr[j] << "', ";
```

```
int i = 0;
while(i < strlen(cstr))
{
    cout << '\'' << cstr[i] << "', ";
    ++i;
}
```

The break and continue Statements

- continue

```
while( cin.get(ch) )
{
     statement1;
     if (ch == '\n')
          continue;
     statement2;
}
statement3;
```

continue skips the rest of the loop body and starts a new iteration

- break

```
while( cin.get(ch) )
{
     statement1;
     if (ch == '\n')
          break;
     statement2;
}
statement3;
```

break skips the rest of the loop and goes to the following statement

The Range-Based for Loop C++11

- Iterates over a collection of elements from the first to the last.
- Can modify a collection by using reference type (will get back to this feature later)

```
double koefs[] = {1.12, 2.13, 3.14, 4.15, 5.16};

for (double x : koefs)
    cout << x << std::endl;



for (int x : {1, 1, 2, 3, 5})
    cout << x << " ";
```

The do .. while Loop (with ~~pre~~ post-condition)

```
statement1;
do
{
    statement2;
} while (test_expr);
statement3;
```

```
// must define outside the loop
char repeatAnswer;

do
{
    // main program
    // ...

    cout << "Press 'Y' to repeat, any other key for exit: ";
    cin >> repeatAnswer;
} while (repeatAnswer == 'y' || repeatAnswer == 'Y');
```