HY150a - Φροντιστηριο 1

07/10/2016

Hello World & std:: namespace

```cpp
#include <iostream>

int main()
{
std::cout<<"Hello World"<<std::endl;
return 0;
}

```

outputs: Hello World

```cpp
#include <iostream>
using namespace std;
int main()
{
cout<<"Hello World"<<endl;
return 0;
}

```

outputs: Hello World

```cpp
#include <iostream>
using namespace std;
int main()
{
std::cout<<"Hello World"<<std::endl;
return 0;
}

```

outputs: Hello World

Hello world in C language

```c
//C hello world example
#include <stdio.h>

int main()
{
  printf("Hello world\n");
  return 0;
}
```

Basic I/O (1)

```
5  cout << "Hello World"<<endl; //Hello World\n
6
7  int num=666;
8  cout << num << endl; //666\n
9
10 cout << "Print this ";
11 cout << "and this!" <<std::endl; //this and this!
12
13 cout << "I like them "<< "together!" <<std::endl;
14 //I like them together!\n
15
16 string str="This is a string";
17 cout<<str<<endl; //This is a string\n
18
19 int age=27; char grade='A';
20 cout << "My ages is "<< age <<" i have grade:"<<grade<<std::endl;
21 //My ages is 27 i have grade:A\n
22
23 float dollars=100.56;
24 cout << dollars<<" Dollars i cash"<<endl; //100.56 Dollars i cash\n
25
```

stream	description
cin	standard input stream
cout	standard output stream
cerr	standard error (output) stream
clog	standard logging (output) stream

http://www.cplusplus.com/doc/tutorial/basic_io/

Basic I/O (2)

```
6
7  int num;
8  cin >> num;// reads integer
9  cout << num << endl;
10
11 int age;
12 char grade;
13 cin >> age >> grade; // reads an integer and a character
14 cout << "My ages is "<< age <<" i have grade:"<<grade<<std::endl;
15
16 float dollars;
17 cin >>dollars; //reads a number as float
18 cout << dollars<<" Dollars i cash"<<endl;
19
```

stream	description
cin	standard input stream
cout	standard output stream
cerr	standard error (output) stream
clog	standard logging (output) stream

http://www.cplusplus.com/doc/tutorial/basic_io/

C-Style I/O

reading integer

```c
#include "stdio.h"

int main(void)
{
   int a;

   printf("Please input an integer value: ");
   scanf("%d", &a);
   printf("You entered: %d\n", a);

   return 0;
}
```

reading string

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

int main(void)
{
  char str[80];

  gets(str);
  printf("Length is %d", strlen(str));

  return 0;
}
```

Reading string from std::cin

cin >> input_string;

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string your_name;
    cout << "Hello, What is your name? ";
    cin >> your_name;
    cout << "Pleased to meet you, " << your_name;
}
```

Program output:

```
Hello, What is your name? firstname lastname
Pleased to meet you, firstname

Press any key to continue . . .
```

getline(cin, input_string);

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    string full_name;
    cout << "Hello, What is your name? ";
    getline( cin, full_name );
    cout << "Pleased to meet you, " << full_name;
}
```

Program output:

```
Hello, What is your name? firstname lastname
Pleased to meet you, firstname lastname

Press any key to continue . . .
```

End Of File (EOF)

Input file:

```
37
77
```

```cpp
#include <iostream>
using namespace std;
int main()
{

int num1;
cin >> num1;// reads integer
cout <<"parsed: "<< num1 << endl;

int num2;
cin >> num2;// reads integer
cout <<"parsed: "<< num2 << endl;

int num3;
cin >> num3;// reads integer
cout <<"parsed: "<<  num3 << endl;

return 0;
}
```

Output :

```
parsed: 37
parsed: 77
parsed: 1413143496
```

```cpp
#include <iostream>
using namespace std;
int main()
{

int read_result;

int num1;
read_result = (cin >> num1);// reads integer
cout <<"parsed: "<< num1 << endl;
cout <<"eof reached "<< (cin.eof()) <<std::endl;
cout <<"read result is: "<< read_result <<std::endl;


int num2;
read_result = (cin >> num2);// reads integer
cout <<"parsed: "<< num2 << endl;
cout <<"eof reached "<< (cin.eof()) <<std::endl;
cout <<"read result is: "<< read_result <<std::endl;

int num3;
read_result = (cin >> num3);// reads integer
cout <<"parsed: "<<  num3 << endl;
cout <<"eof reached "<< (cin.eof()) <<std::endl;
cout <<"read result is: "<< read_result <<std::endl;

return 0;
}
```

Output :

```
parsed: 37
eof reached 0
read result is: 1
parsed: 77
eof reached 0
read result is: 1
parsed: 0
eof reached 1
read result is: 0
```

ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

```
char symbol;
std::cin >> symbol;
std::cout<<(int) symbol<<std::endl;
```

Printing the decimal representation of character

```
int nums;
std::cin >> nums;
std::cout << (char) nums<< std::endl;
```

Printing the the ASCII character which the number corresponds

Unix File Redirection

- To run prog1 but read data from file infile instead of the keyboard, you would type

  `prog1 < infile`

- To run prog1 and write data to outfile instead of the screen, you would type

  `prog1 > outfile`

- Both can also be combined as in

  `prog1 < infile > outfile`

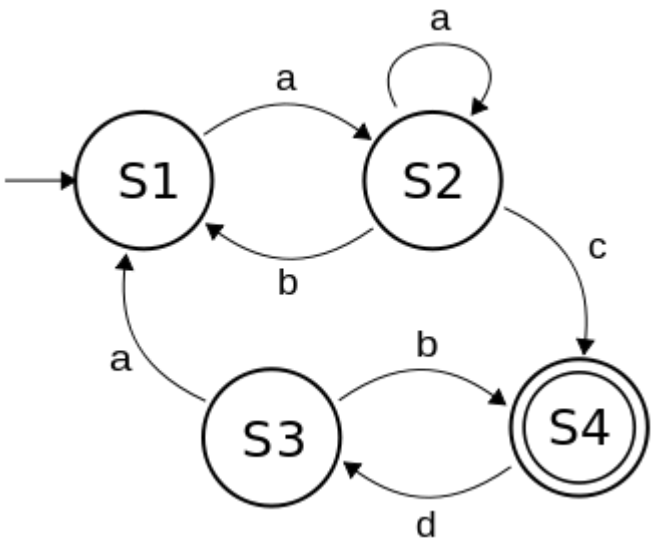
- Redirection is simple, and allows a single program to read or write data to or from files or the screen and keyboard.

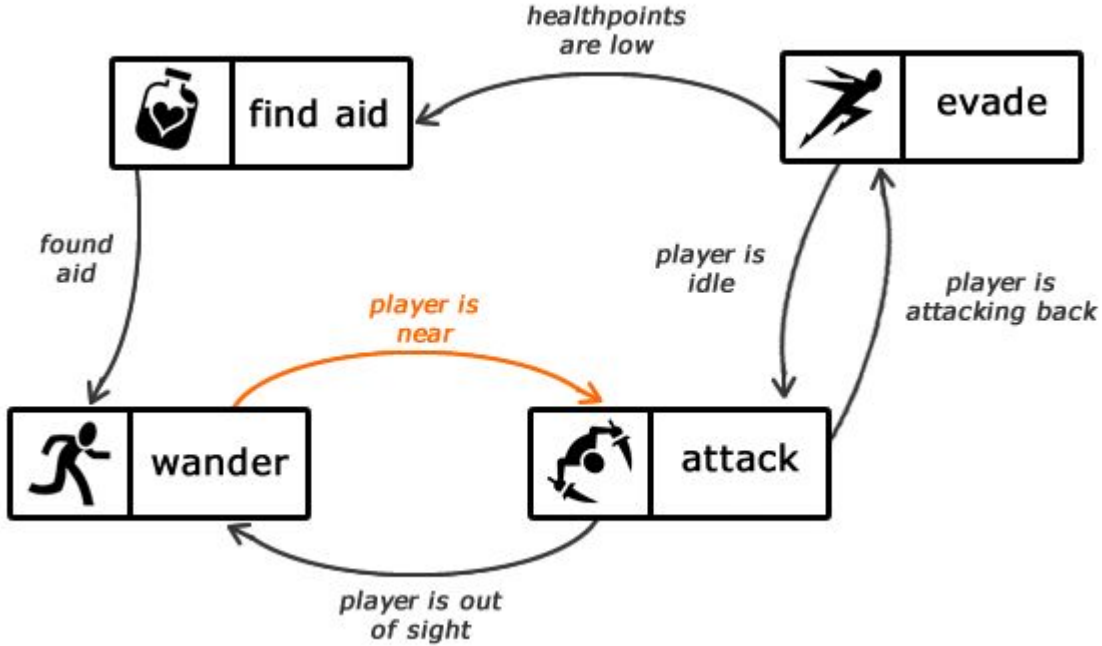
Compile and run with file redirection.
g++ test.cpp -o myapp
./myapp < testfile.txt

http://www.cplusplus.com/doc/tutorial/basic_io/

FSM -Finite State Machines

```
1
2  enum MyState {STATE_0,STATE_1,STATE_0,STATE_END};
3  Mystate state=STATE_0;
4
```





FSM - Finite State Machines

```
3  #define STATE_0   0
4  #define STATE_1   1
5  #define STATE_2   2
6  #define STATE_END 3
7
8  int main(){
9  int state = STATE_0;
10 while (state != STATE_END)
11 {
12     switch (state)
13     {
14       case STATE_0:
15         std::cout << "state 0 here!" << std::endl;
16         state=STATE_2;
17         break;
18       case STATE_1:
19         std::cout << "state 1 here!" << std::endl;
20         state = STATE_END;
21         break;
22       case STATE_2:
23         std::cout << "state 2 here!" << std::endl;
24         state = STATE_1;
25         break;
26       default:
27         std::cerr << " ERROR STATE " << std::endl;
28         return -1;
29         break;
30     }
31 }
```

default case is invoked if all checks fails

```
3  #define STATE_0   0
4  #define STATE_1   1
5  #define STATE_2   2
6  #define STATE_END 3
7  int main()
8  {
9  int state = STATE_0;
10 while (state != STATE_END)
11 {
12     switch (state)
13     {
14       case STATE_0:
15         std::cout << "state 0 here!" << std::endl;
16         state=STATE_2;
17       case STATE_1:
18         std::cout << "state 1 here!" << std::endl;
19         state = STATE_END;
20       case STATE_2:
21         std::cout << "state 2 here!" << std::endl;
22         state = STATE_1;
23       default:
24         std::cerr << " ERROR STATE " << std::endl;
25         return -1;
26     }
27 }
28
```

Forgetting **break** statement will cause bugs!

How to compile and execute a c++ program on linux

- **Compiling the a.out**
  g++ my_program.cpp
  ./a.out

- **Renaming the executable**
  g++ my_program.cpp -o AppName
  ./AppName

- If you should happen to get permission errors, you need to make the file executable. You can do this by issuing the following commands below

```
chmod +x App
```

Finding material and c++ tutorials

- Read the slides!
- Google!
- man pages
- books!
- http://www.cprogramming.com/tutorial/references.html
- http://www.cplusplus.com/reference/
- http://stackoverflow.com/

Questions?

Extra slides

Run Length Encoding

- Run length encoding (RLE) απλούστερη μέθοδος συμπίεσης.
- Κωδικοποιεί πληροφορία για την επανάληψη των χαρακτήρων / στοιχείων εισόδου
- Είναι ο αλγόριθμος που χρησιμοποιεί το FAX (τηλεομοιοτυπία)

RLE

- As a basic example, consider the following string of numbers:

5 5 5 5 8 8 8 2 2 2 2 2

- There is a fair amount of redundancy there. In RLE notation, this same string could be expressed as:

4 5 3 8 5 2

RLE

1 2 3 4 5 6

- Apply the same RLE compression scheme as before:

1 1 1 2 1 3 1 4 1 5 1 6

Παράδειγμα

WWWWWWWWWWWWBWWWWWWWWWWWWWWBBBWWW
WWWWWWWWWWWWWWWWWWWWWWWWBWWWWWW
WWWWWWWW

12W1B12W3B24W1B14W

Πως ψάχνουμε για εντολές βιβλιοθήκης

Εντολή συστήματος “man”

- If you already know the name of the function you want, you can read the page by typing (to find about strcat).

> man 3 strcat

- If you don't know the name of the function, a full list is included in the introductory page for section 3 of the manual. To read this, type

> man 3 intro

- There are approximately 700 functions described here. This number tends to increase with each upgrade of the system. On any manual page, the SYNOPSIS section will include information on the use of the function. For example

#include <time.h> char *ctime(time_t *clock)

- This means that you must have #include <time.h> in your file before you call ctime. And that function ctime takes a pointer to type time_t as an argument, and returns a string (char *). time_t will probably be defined in the same manual page. The DESCRIPTION section will then give a short description of what the function does. For example

ctime() converts a long integer, pointed to by clock, to a 26-character string of the form produced by asctime().

Οργάνωση κώδικα

- Σε μεγαλύτερα προγράμματα, οι δηλώσεις των συναρτήσεων συγκεντρώνονται σε ένα ή περισσότερα header files (κατάληξη **.h**), τα οποία συμπεριλαμβάνονται κατά την προεπεξεργασία του κώδικα:

```
#include "name.h"
```

- όπου name.h το όνομα του header file και μπορεί να περιλαμβάνει και το path.
- Οι headers που ορίζει ο προγραμματιστής περικλείονται σε “”.
- Οι headers του συστήματος περικλείονται σε <>.
- Με την συμπερίληψη των κατάλληλων headers:
  - ο compiler γνωρίζει τον τρόπο κλήσης των συναρτήσεων ή εκτελέσιμων βιβλιοθηκών που χρειάζεται ένα τμήμα κώδικα.
  - ο προγραμματιστής μπορεί να επισκοπήσει εύκολα τη λειτουργικότητα του κώδικα