C/C++ for Matlabbers

Ahmet Sacan

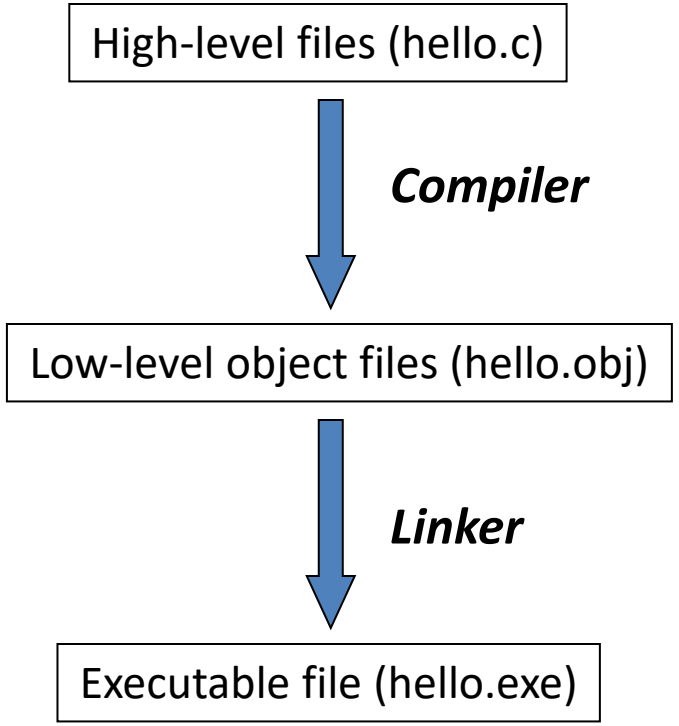
Some slides are from: Ahmet's 20013/230teaching

a sample program: add2.cc

```
/* by AhmetSacan.
summary: asks user for two numbers. prints the sum of these numbers.
*/
// include header files where the functions you want are defined.
#include <stdio.h>

int main(){
 // declare variables
 float f1,f2;
 float sum;

 /*read values from input*/
 scanf("%f %f", &f1, &f2);

 /*do needed calculations...*/
 sum = f1 + f2;

 /*print out the desired results.*/
 printf("%f", sum);

 return 0;
}
```

Writing & Running C/C++

- You need an editor software
  - can be any editor..
- and a compiler/Linker:
  - Visual Studio
  - gcc/g++
  - xcode
  - Clang
  - mingw
  - Intel's compiler
  - others.. http://www.stroustrup.com/compilers.html



Types of Statements

- Declarative

```
int x;
```

- Assertive

```
printf("%f", sqrt(5) + 7);
x = y / 3;
```

- Conditional (if, if-else, case)

```
if(x > y)
```

- Controlling (while, do-while, for, goto)

```
while(x<10)
```

Data-type	Size	Range
char	1 bytes	-128 to 127
unsigned char	1 bytes	0 to 255
short	2 bytes	-32768 to 32767
unsigned short	2 bytes	0 to 65535
int	4 bytes	-2147483648 to 2147483647
unsigned int	4 bytes	0 to 4294967295
long	8 bytes	-2147483648 to 2147483647
unsigned long	8 bytes	0 to 4294967295
float (6 digits)	4 bytes	1.175494e-38 to 3.402823e+38
double (15 digits)	8 bytes	2.225074e-308 to 1.797693e+308

Type Conversion

- conversions among different data types may occur while evaluating an expression.

```
int x=3.4, y=4;
float z = x;
char c = z;
```

- Two type of conversions:
  - Implicit (automatic)
  - explicit (type-cast)

Beware of implicit type conversions

```
short s; int i; long l;
float f; double d;
```

- Safe conversion (to broader type)

```
l=i; l=s; i=s;
d=i; f=i; /*100 converted to 100.0
d=f; /*?*/
```

- Unsafe conversion (loosing information)

```
i=f; /* truncation: 3.14 to 3 */
s=l; /* dropping bits */
f=d; /* truncation/rounding */
```

- Probably unsafe operation

```
int x=5, y=2;
float f = x/y;
```

Explicit Conversion (type-cast)

- Most compilers give warnings for possibly unsafe conversions. One type of data can be explicitly forced into another data type.

```
float f=4.3; int x, y=2;
x=3.7; /*compiler warning*/
x=(int) f; /*ok*/
f = x/(float)y;
```

Selection/loop blocks

- The conditions must be enclosed in parenthesis.
- There is no "end" keyword.
- Blocks of statements are enclosed by {....}
  - When a single statement, {...} is optional (but recommended).

Find the output of the following:

```
int x=3;
if(x != 3)
  printf("t");
  printf("a");
if(x % 2){
  printf("hm");
  if(x>0){
   printf("e");
  }
}
if(x-3)
  if(x==3)
    printf("s");
    printf("t");


if(0); { printf(" sac"); printf("an"); }
```

Exercise

- Write a program multiconvert.cc that reads from the user, a conversion code and a number; and converts the given number according to the following conversion rules:

code	Conversion Type	Conversion Rule
f	From Fahrenheit(F) to Celcius(C)	C=5/9*(F-32)
i	From Inches(I) to centimeters(cm)	cm=3/2*I
m	From miles(M) to kilometers(Km)	Km=1.6*M
p	From pound(P) to kilograms(Kg)	Kg=0.45*P

>> multiconvert.exe
20 p
9 .000000

gim'me a break.

- In a switch statement: if "break;" statement is not used, the execution continues down until a "break;" is encountered.

```
x=2;
switch(x){
   case 1 : printf("one"); break;
   case 2 : printf("two");
   case 3 : printf("three"); break;
   case 4 : printf("four");
 }
```

Syntactic candy

- a = b ? c : d;
- a += 5;
- a++;
- a = b = 5;
- a + (b=5)
- if (a = 2){ … }
- Define your own macros using #define
  - Macro: #define A B
  - "Whenever I write A, let's pretend I wrote B."

Loops

- for (<expr1> ; <test> ; <expr2>) { … }
- while(<test>) { … }
- do { … } while(<test>);

Control Statements

- break; exits a loop-control.
- continue; continues the next iteration of a loop-control.
- return [<arg>]; exits a function.
- exit(int); exits the program.

Function Definition

```
<return-type> <function-name>
              ( <parameter-list> )
{
  ...
  <statements>
  ...
}
```

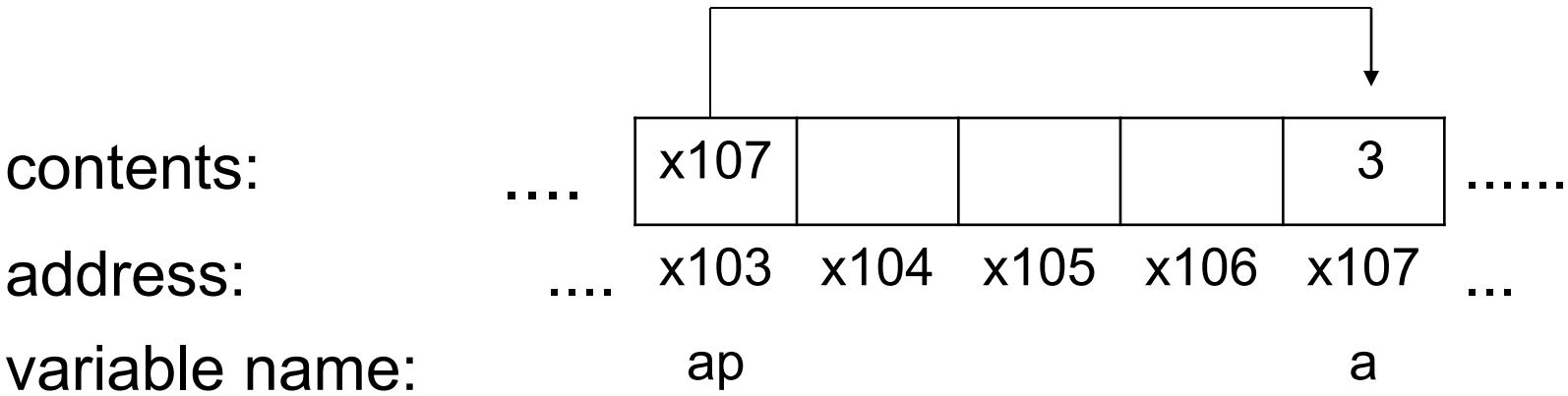
function head

function body

Function Types

- with / without return-type.
  - void PrintIntro( );
  - int ReadAnInt( );
- with / without arguments.
  - void PrintResult(int res);
  - void PrintIntro( );
- type of arguments: readable, writeable...
    - int Add(int a, int b);
  - void Add(int a, int b, int * sum);
  - void Add(int a, int b, int & sum);

Pointers



ap	*ap	a	&a
x107	3	3	x107

Pointer declaration

- <type> * <varName> ;

```
int iVar;         /* declares an int. */
int * iVarPtr;    /* declares a variable that
                     will POINT to an integer */
int * ptr2;

float f;
float * myFloatPointer;
```

Pointer Assignment

- &<varName> /* address of <varName> */

```
iVarPtr = &iVar;
ptr2 = iVarPtr;

myFloatPointer = &f;

iVarPtr = iVar;                  /* BAD... */
myFloatPointer = iVarPtr;     /* INVALID */
```

```
int Add(int x, int y){ return x+y; }
void main(){
 ...
 c = Add(a, b);
...
```

```
void Add(int x, int y, int *z) { *z =  x+y; }
void main(){
 ...
 Add(a, b, &c);
...
```

Static Arrays

- <type> <arrayName> [expr-1]...[expr-n]
  - expr-i must be a constant integral type.
- **Array indices start from 0.**

```
int midtermScores[50];

#define NUM_EXAMS 3

float exams[NUM_EXAMS];

int x = 10;
float prices[x];     /* INVALID!
                        array-expr must be constant */
```

Dynamic Arrays

- C: malloc/free

```
int *a;
a = (int*) malloc(sizeof(int) * 10);
... a[1]=5; ...
free(a)
```

- C++:

```
int *a;
a = new int[10];
... a[1]=5; ...
delete[] a;
```

Copying Arrays

- an array cannot be directly assigned to another array.

```
int x[100], y[100];
x = y;       /* ILLEGAL */
```

- You must copy values one-by-one:

```
for(i=0; i<100; i++)
  x[i] = y[i] ;
```

Command-line arguments

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

int main(int argc, char **argv) {
        if (argc != 3) {
                printf("usage: %s x y\n", argv[0]);
                exit(2);
        }
        float x; float y;
        x = atof(argv[1]);
        y = atof(argv[2]);
        printf("%f\n", x + y);
        return 0;
}
```

Exercise

- Write a function multiconvert.m that takes two command line arguments: a conversion code and a number; and converts the given number according to the following conversion rules:

code	Conversion Type	Conversion Rule
f	From Fahrenheit(F) to Celcius(C)	C=5/9*(F-32)
i	From Inches(I) to centimeters(cm)	cm=3/2*I
m	From miles(M) to kilometers(Km)	Km=1.6*M
p	From pound(P) to kilograms(Kg)	Kg=0.45*P

>> multiconvert.exe 20 p
9 .000000

#include <math.h>

sin(x)	**sine of x**
cos(x)	**cosine of x**
tan(x)	**tan of x**
asin(x)	**arcsine of x, result between -pi/2 and +pi/2**
acos(x)	**arccosine of x, result between 0 and +pi**
atan(x)	**arctan of x, result between -pi/2 and +pi/2**
atan2(y,x)	**arctan of (y/x), result between -pi and +pi**
hsin(x)	**hyperbolic sine of x**
hcos(x)	**hyperbolic cosine of x**
htan(x)	**hyperbolic tan of x**
exp(x)	**exponential function**

#include <math.h> (cont'd)

log(x)	natural logarithm
log10(x)	logarithm to base 10
pow(x,y)	x to the power of y (x^y)
sqrt(x)	the square root of x (x must not be negative)
ceil(x)	ceiling; the smallest integer not less than x
floor(x)	floor; the largest integer not greater than x
fabs(x)	absolute value of x
ldexp(x,n)	x times 2**n
frexp(x, int *exp)	returns x normalized between 0.5 and 1; the exponent of 2 is in *exp
modf(x, double *ip)	returns the fractional part of x; the integral part goes to *ip
fmod(x,y)	returns the floating-point remainder of x/y, with the sign of x

<ctype.h> Character Testing.

- int isalnum(int c) -- True if c is alphanumeric.
- int isalpha(int c) -- True if c is a letter.
- int isascii(int c) -- True if c is ASCII .
- int iscntrl(int c) -- True if c is a control character.
- int isdigit(int c) -- True if c is a decimal digit
- int isgraph(int c) -- True if c is a graphical character.
- int islower(int c) -- True if c is a lowercase letter
- int isprint(int c) -- True if c is a printable character
- int ispunct (int c) -- True if c is a punctuation character.
- int isspace(int c) -- True if c is a space character (space, newline or a tab).
- int isupper(int c) -- True if c is an uppercase letter.
- int isxdigit(int c) -- True if c is a hexadecimal digit

<ctype.h> Character Conversion

- int toascii(int c) -- Convert c to ASCII .
- int tolower(int c) -- Convert c to lowercase.
- int toupper(int c) -- Convert c to uppercase.

Strings

- In C, use a char array for string. Convention is to end the string with a "null" character.
  - You always need one extra null character at the end of the array.
  - Difficult to maintain the array that will always have enough elements to store the string you need.
- In C++, the string class will auto-size for you as needed.

C-string and string class functions

- C-String Functions
  - strcpy(str1, str2) Copies str2 to str1.
  - strlen(str) Returns the number of characters in str.
  - strcat(str1, str2) Concatanates (joins) str2 to str1.
  - strcmp(str1, str2) alphabetically compares str1 to str2 (return 0 if equal).
- String Class Functions
  - str.length() Returns the number of characters in str.
  - str[i] Get ith character.
  - str1.find(*str2*) Find first occurrence of str2, return int position.
  - str1.replace(pos, n, char) Replace n characters in str1 with char x starting at pos.
  - str1.append(str2) Appends str2 to str1.
  - str1.compare(str2) Compares str1 to str2. (return 0 if equal)
  - str1.substr(pos, n) Return n characters in str1 starting at pos.

Classes in C++

```
struct person
{
  string name;
  int age;
};
```

```
class person
{
  public:
    string name;
    int age;
};
```

Exercise

- Implement and demonstrate the person and student classes that we developed in Matlab.

Many libraries available...

- https://github.com/fffaraz/awesome-cpp