Biostatistics 615 - Statistical Computing

Lecture 2
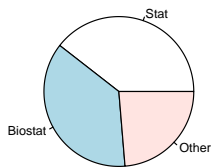## Computer Arithmetic & Basic Syntax
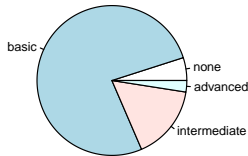
Jian Kang

September 10, 2015

Summary of the previous lecture

- ✓ Syllabus
- ✓ Introduction to statistical computing
- ✓ Introduction to computer representation of data
  - ✓ Binary representation
  - ✓ Storage unit (Bit, Byte, KB, MB, GB, TB, PB)
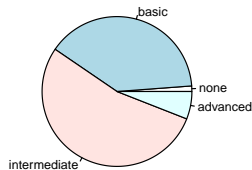- ✓ Class survey

Storage in C++

- How the ideas about number representation are implemented in C++?

- While investigating this issue, we will also introduce some of language features that will allow us to begin to write, compile and execute C++ programs

- The amount of storage that is allocated to different data types can be determined using the sizeof operator

- Example 2.1: demonstrates the use of sizeof to determine information about storage space, in bytes

```cpp
//Example 2.1: storageSize.cpp
#include <iostream>
using namespace std;
int main(){
  cout << "The storage allocated for a char is "
  << sizeof(char) << " byte" << endl;
  cout << "The storage allocated for an unsigned short integer is "
  << sizeof(unsigned short int) << " bytes" << endl;
  cout << "The storage allocated for an integer is "
  << sizeof(int) << " bytes" << endl;
  cout << "The storage allocated for a long integer is "
  << sizeof(long int) << " bytes" << endl;
  cout << "The storage allocated for an unsigned long integer is "
  << sizeof(unsigned long int) << " bytes" << endl;
  cout << "The storage allocated for a float is "
  << sizeof(float) << " bytes" << endl;
  cout << "The storage allocated for a double is "
  << sizeof(double) << " bytes" << endl;
  cout << "The storage allocated for a long double is "
  << sizeof(long double) << " bytes" << endl;
  return 0;
}
```

Basic syntax – comment statement

```
//Example 2.1: storageSize.cpp
... ...
```

- Use "//" at the start of each line

- Bracket the comment encompassing possibly multiple lines by "/∗" and "∗/"

Basic syntax – header files

```
//Example 2.1: storageSize.cpp
#include <iostream>
... ...
```

- #include<...>: directive to the preprocessor to include functions and classes in the C++ Standard Library, where the code has been compiled and will be automatically linked with the developed program
- iostream: input/output operator classes
- Many other tools are available: e.g.
  - cmath: many the standard math functions
  - array: container class template for a fixed size array

Basic syntax – statement "using"

```
//Example 2.1: storageSize.cpp
#include <iostream>
using namespace std;
... ...
```

- This ensures that there will be no ambiguity in referring to functions and classes in the C++ Standard Library

- A *namespace* is a collection of definitions of variables, functions and other key components associated with a library or program

- Example 2.1 uses a cout object and the output manipulator endl that are declared in the iostream header file.

```
cout << "The storage allocated for a char is " << ... << endl;
```

- *scope resolution operator* "::" to bypass the using directive

```
std::cout << "The storage allocated for a char is " << ... << std::endl;
```

Basic syntax – main function

```
//Example 2.1: storageSize.cpp
#include <iostream>
using namespace std;
int main(){...}
```

- Every C++ program has one and only one main function.

- It directs the flow activity within a program.

- Return type: int.

- Parameters
    - must be enclosed by a pair of parentheses "(" and ")"
    - can be void, two-parameter form or multi-parameter form

- Body
    - must be enclosed by the matching curly braces "{" and "}"

- In Example 2.1, the main is just a collection of explanatory text to be printed with values that are returned by the sizeof operator.

Basic syntax – output operator

```
//Example 2.1: storageSize.cpp
#include <iostream>
using namespace std;
int main(){
  cout << "The storage allocated for a char is "
  << sizeof(char) << " byte" << endl;
  ...
  return 0;
}
```

- Output insertion operator: "<<" for standard output object *cout*

- Output manipulator: ``endl"
    - appears at the end of each output line
    - produces a carriage return (starts a new line)
    - flushes (cleans out) the output buffer

Basic syntax – statement "return"

```cpp
//Example 2.1: storageSize.cpp
#include <iostream>
using namespace std;
int main(){
  cout << "The storage allocated for a char is "
  << sizeof(char) << " byte" << endl;
  ...
  return 0;
}
```

- In general, it returns control back to a calling function

- In main function, it transfers control back to the operating system

- The main function may not have the return statement, in which case "return 0;" will be executed

Compile

- The "storage size" source code was saved as "storageSize.cpp"

- The cpp file extension is one of the permissible options for the GNU C++ compilier that is available in Unix environments

- To run the program, we first compile it using

```
$ g++ -Wall storageSize.cpp -o storageSize
```

- It transforms the input file storageSize.cpp into machine language

  - "g++" invokes the GNU compiler
  - "-Wall" turns on all of the most commonly used compiler warnings
  - "-o" specifies the name of the executable program

Run

- To load and run the compiled program, the name of the executable program (i.e., storageSize) is entered on the command line prefaced by the "./" modifier that informs the shell where to look for the executable.

```
$ ./storageSize
The storage allocated for a char is 1 byte
The storage allocated for an unsigned short integer is 2 bytes
The storage allocated for an integer is 4 bytes
The storage allocated for a long integer is 8 bytes
The storage allocated for an unsigned long integer is 8 bytes
The storage allocated for a float is 4 bytes
The storage allocated for a double is 8 bytes
The storage allocated for a long double is 16 bytes
```

Complete list of fundamental types

Group	Type names*	Notes on size / precision
Character types	char	Exactly one byte in size. At least 8 bits.
	char16_t	Not smaller than char. At least 16 bits.
	char32_t	Not smaller than char16_t. At least 32 bits.
	wchar_t	Can represent the largest supported character set.
Integer types (signed)	signed char	Same size as char. At least 8 bits.
	signed short int	Not smaller than char. At least 16 bits.
	signed int	Not smaller than short. At least 16 bits.
	signed long int	Not smaller than int. At least 32 bits.
	signed long long int	Not smaller than long. At least 64 bits.
Integer types (unsigned)	unsigned char	(same size as their signed counterparts)
	unsigned short int	
	unsigned int	
	unsigned long int	
	unsigned long long int	
Floating-point types	float	
	double	Precision not less than float
	long double	Precision not less than double
Boolean type	bool	
Void type	void	no storage
Null pointer	decltype(nullptr)	

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

Nonnegative integer storage in C++

- Take unsigned short data type as an example.
- How many bytes of storage?
- What is the range?
- How to write a C++ program to obtain the binary representation of an unsigned short integer?
  - The number can be viewed as having ___ possible slots corresponding to 2^0 all the way to ___ .
  - The program can step through each of these slots

Binary representation of nonnegative integer

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){
  unsigned short inVal;
  cout << "Enter an integer between 0 and 65535: ";
  cin >> inVal;
  cout << "This integer in binary is ";
  printBinary(inVal);
  cout << endl;
  return 0;
}
void printBinary(unsigned short val){
  for (int i = 15; i >=0; i--)
      if(val & (1 << i))
        cout << "1";
      else
        cout << "0";
}
```

Basic syntax – prototype or declaration of function

```cpp
//Example 2.2: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){...}
void printBinary(unsigned short val){...}
```

- Example 2.2 employs some elementary bit-wise operations in a function called printBinary to pick off the internal binary representations of a nonnegative integer

- The *prototype* or *declaration* of the basic form for printBinary that appears prior to the main function

- The *return type* of the function is designated as void which means it does not return a value to the calling program

- In C++, it is necessary to tell the compiler about the essential details (e.g. return type and type for arguments) in a function before it can be used

Basic syntax – Type declaration

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){
   unsigned short inVal;
...
}
void printBinary(unsigned short val){...}
```

- The variable inVal will hold the integer whose binary representation will be determined
- The code consists of a *type declaration* which states that inVal is of storage type unsigned short
- In C++, every variable's type must be explicitly stated when it is introduced into program

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){
  unsigned short inVal;
  cout << "Enter an integer between 0 and 65535: ";
  cin >> inVal;
  printBinary(inVal);
  cout << endl;
  return 0;
}
void printBinary(unsigned short val){...}
```

- cin is the standard input object

- >> is the input operator

- The code reads a value from the shell command line into inVal

Basic syntax – The for loop

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){...}
void printBinary(unsigned short val){
  for (int i = 15; i >=0; i--)
      if(val & (1 << i))
        cout << "1";
      else
        cout << "0";
}
```

- int i is the type declaration of the index variable i
- Three important things in a for loop for the index variable i
    1. Starting value ($i = 15$)
    2. Logical condition (False) to stop the loop ($i >=0$)
    3. Rule to move (i - - or $i = i - 1$)

Basic syntax – Bit shift operators

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){...}
void printBinary(unsigned short val){
  for (int i = 15; i >=0; i--)
      if(val & (1 << i))
        cout << "1";
      else
        cout << "0";
}
```

- "$<<$" is the left bit shift operator
- "$a << k$" cause the bits in `a` to be shifted left by `k` positions
  - If $a = 15 = (00001111)_2$, then $(a << 1) = 30 = (00011110)_2$
  - "$a << k$" is equal to "$a*\underbrace{2 * ... * 2}_{k}$"; Thus, "$1 << i$" $= 2^i$.

- Recall "$<<$" is also the output operator with `cout` (Operator overloading)

Basic syntax – Bit-wise AND operator

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){...}
void printBinary(unsigned short val){
  for (int i = 15; i >=0; i--)
      if(val & (1 << i))
        cout << "1";
      else
        cout << "0";
}
```

- "`&`" is the bit-wise AND operator

- "`a & b`" each bit of `a` to the corresponding bit of `b`. If both bits are 1, the corresponding result bit is set to 1. Otherwise, the corresponding result bit is set to 0.

  - If `a` = 15 = $(00001111)_2$ and `b` = 16 = $(00010000)_2$, then "`(a & b)` = $(00000000)_2$ = 0"

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){...}
void printBinary(unsigned short val){
  for (int i = 15; i >=0; i--)
      if(val & (1 << i))
        cout << "1";
      else
        cout << "0";
}
```

- If `a` $> 2^i$ and `a` $< 2^{i+1}$, then
  (`a` & 2^i) > 0, otherwise (`a` & 2^i) = 0
- Why is this true?

Basic syntax – Statement if ... else ...

```cpp
//Example 2.1: nonNegIntBinaryRep.cpp
#include <iostream>
using namespace std;
void printBinary(unsigned short val);
int main(){...}
void printBinary(unsigned short val){
  for (int i = 15; i >=0; i--){
      if(val & (1 << i)){
        cout << "1";
      }
      else{
        cout << "0";
      }
   }
}
```

- Three parts: 1) *boolean expression*: `val & (1 << i)`; 2) *If block of code*: `cout << "1";`; 3) *else block of code*: `cout << "0";`
- If the *boolean expression* is `true`, then the *if block of code* will be executed, otherwise *else block of code* will be executed

Environment for homework : Connect to scs.itd.umich.edu

- Windows environment : Use PuTTY for command line and WinSCP for file transfer

- Mac or Linux

  Command line ssh [your-uniqname]@scs.itd.umich.edu
  File transfer scp [your-file] [your-uniqname]@scs.itd.umich.edu:[path-to-destination]

Tips :

- See http://www.itcs.umich.edu/scs/ and http://www.itcs.umich.edu/ssh/ for details

- Linux tutorial: http://ryanstutorials.net/linuxtutorial/

- Add the following line in ~/.cshrc file for more convenient command line (which shows current working directory).

  ```
  set prompt="`whoami`@`hostname -s`:%~$ "
  ```

Steps for Homework 0

1. Create a directory `Private/biostat615/hw0/` in your home directory
   - Create a directory `Private/biostat615/hw0/` using WinSCP
   - Type `mkdir -p ~/Private/biostat615/hw0/` in the command line
   - Make sure your homework is in private space. If it is accessible by someone else, your homework will be discarded.

2. Create a file (e.g. `Private/biostat615/hw0/storageSize.cpp`)
   - Directly type `vi ~/Private/biostat615/hw0/storageSize.cpp`
   - Copy a file remotely using `WinSCP` or `scp`

3. Use basic Unix commands (e.g. `cd ~/my/path/`, `pwd`) to navigate between directories.

4. Compile and run the program

5. Before submission, remove all the executable and object (.o) files, and compress your code (under Private/biostat615)

```
$ cd ~/Private/biostat615/hw0/
$ rm storageSize
$ cd ../
$ tar czvf hw0.tar.gz hw0/
```

Homework 0

- Implement the following two programs and submit to the instructor by copying the compressed source codes to
  ~jiankang/biostat615/[your-uniqname]/
  storageSize.cpp
  nonNegIntBinaryRep.cpp
- This homework will not be graded, but mandatory to submit for everyone who wants to take the class for credit
- No due date, but homework 0 must be submitted prior to submitting any other homework.

Steps for homework 0

1. ssh [your-uniqname]@scs.itd.umich.edu
2. mkdir -p ~/Private/biostat615/hw0/
3. cd ~/Private/biostat615/hw0/
4. vi storageSize.cpp
5. (input the code)
6. rm storageSize
7. vi nonNegIntBinaryRep.cpp
8. (input the code)
9. rm nonNegIntBinaryRep
10. cd ../
11. tar czvf hw0.tar.gz hw0/
12. cp hw0.tar.gz ~jiankang/biostat615/[your-uniqname]/

Recourses

- C++ Tutorials: http://www.cplusplus.com/doc/tutorial/
- C++ IDE on local computers
    - Code::blocks: Windows/Linux/MacOS,
      http://sourceforge.net/projects/codeblocks/
    - Visual studio express: Windows,
      https://www.visualstudio.com/en-US/products/visual-studio-express-vs
    - Vi: Linux/Mac OS
      http://www.unix-manuals.com/tutorials/vi/vi-in-10-1.html
      http://www.thegeekstuff.com/2009/01/tutorial-make-vim-as-your-cc-ide-using-cvim-plugin/

Summary

- Datatypes in C++
- Binary representation of a nonnegative integer in C++
- Basic syntax
  - Comment statement
  - Header files (iostream)
  - Statement "using"
  - The main function (return type and return statement)
  - Output/input operators (cin, cout, << , >>)
  - Prototype / declaration of functions and variables
  - The "for" loop and statement "if ... else ... "
  - Bit-wise operators (shift and AND)
- Compile and run using GNU C++ complier
- Environment for homework
- Homework 0
- Resources