EE2331 Data Structures and Algorithms

Introduction

Intended Learning Outcomes

- Students should be able to
  - apply structural programming approach to solve more complex computation problems;
  - demonstrate applications of standard data structures such as linked list, stack, queue and tree;
  - solve computation problems using recursion where appropriate;
  - understand different sorting and searching algorithms.

Tentative Syllabus

- Review basic concepts of C++
- Computation Complexity (Big-O)
- Linked Lists
- Stacks and Queues
- Recursion
- Binary Trees
- Sorting and Searching Algorithms
- Hashing

Assessment

- Exam 60%
- Coursework 40%
    - Programming Assignments – 20%
    - Test – 10%
    - Participation – 10%
        - Submission of tutorial problems
        - Submission of in-class exercises
        - Attendance

- Copying of assignments is strictly prohibited. If the submitted program of student *A* is found to be the “same” as that of student *B*, both students *A* and *B* will receive zero mark for that assignment.

Teaching Plan

- 2-hour lecture
  - Go through the lecture notes and elaborate concepts

- 1-hour lecture
  - Review/introduce tutorial solutions/problems
- 2-hour tutorial
  - Intensify your concepts via coding practice
  - Finish and demonstrate your implementation in class

General Opinions of Students

- Student:
  - At the beginning I am comfortable with the materials. But I skip a class in week 3, and then I don’t quite understand the discussion for the rest of the semester.
- Teacher:
  - Materials discussed in this course are highly correlated. So, **DON’T SKIP** any classes.

General Opinions of Students

- Student:
  - I have good grade in CS2363/CS2311. But I find EE2331 a lot more difficult.
- Teacher:
  - EE2331 is a very basic course in computing. It is not difficult. But you have to adjust your learning method. We emphasize on problem solving rather than syntax.
  - A computer program cannot be constructed by copy-and-paste. You have to understand the methods, and be able to apply what you have learnt in a different problem setting. It is counter-productive in trying to memorize the program codes.

General Opinions of Students

- Student:
  - I understand the materials presented by the instructor in lectures. But I just can’t write my own programs.
- Teacher:
  - There are different levels of understanding. You have to be able to generalize the example solutions discussed in class to solve other problems.
  - You must acquire the following skills
    - how to formulate a solution method to solve a problem
    - how to organize and express your ideas in a systematic manner
    - how to translate your ideas to program codes
    - the ability to trace the execution of program codes (without this skill, you can never find out why your program doesn’t work)
    - these skills cannot be learnt by just memorizing the notes!!

General Opinions of Students

- Student:
  - My classmate(s) can finish the tutorials or assignments very quickly. But I have spent many hours or even days in front of the computer and cannot get my program working.
- Teacher:
  - Let’s take “swimming” as an analogy. You can memorize the notes on “how to swim” in a couple of minutes. But you cannot learn how to swim by this way.
  - Programming skills can only be acquired via practices. **NO PAIN, NO GAIN.**
  - You have to
    - pay attention in class
    - follow the guidelines given to you
    - do the programming exercises yourself
    - learn from your own mistakes
    - read extra references whenever you need (Google is your friend!)
    - do not take the short-cut

Course Outline of EE2331

INTRODUCTION

Why Data Structures and Algorithms?

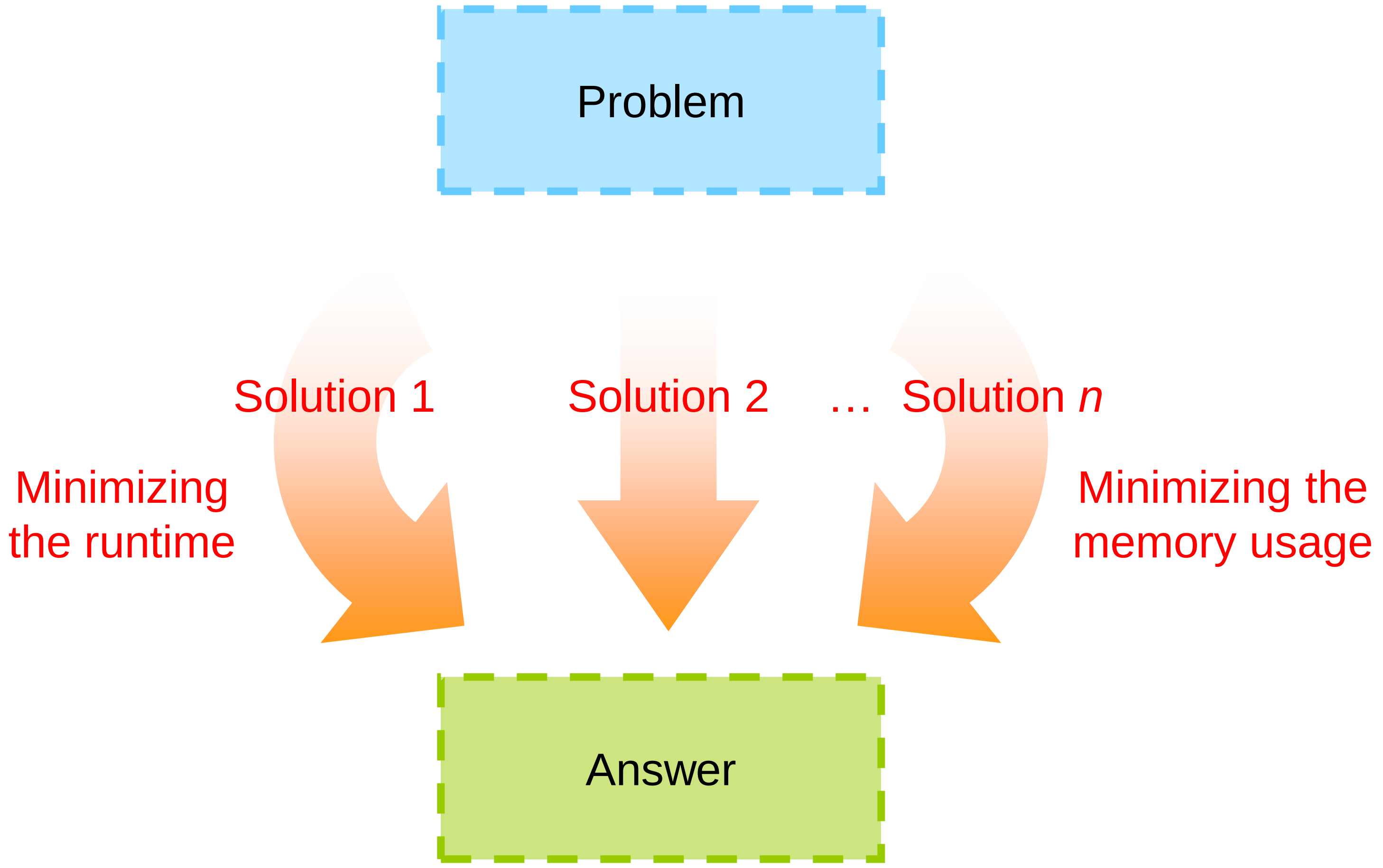
- All about problem solving
  - We don’t want to get success by chance
  - Problems once solved, should be able to be solved again
  - We also want to apply the same solution to solve similar problems. No one wants to reinvent the wheels.
  - A generic solution is needed

- We need to solve problems in a systematic, structural, repeatable and extensible way

Data Structure + Algorithm

= Program

Multiple Solutions to a Problem



Algorithms

- Methods for solving problems which are suited for computer implementation
- In writing a computer program, one is generally implementing a method of solving a problem
- This method is often independent of the particular computer to be used
- It is the method, not the computer program itself, to be studied to learn how the problem is being attacked

Data Structures

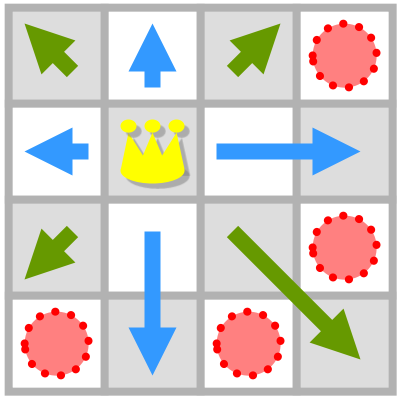
- Most algorithms involve complicated methods of organizing the data involved in the computation
- Data structures are describing the way how the computer store, and organize the data, as well as how the data is accessed.

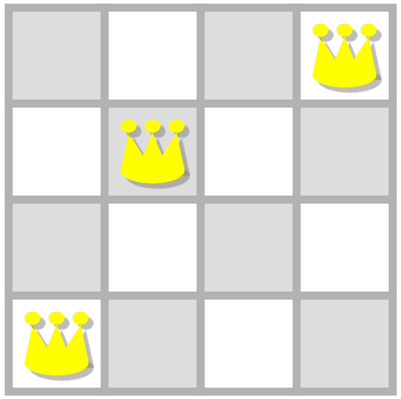
Data Structures + Algorithms

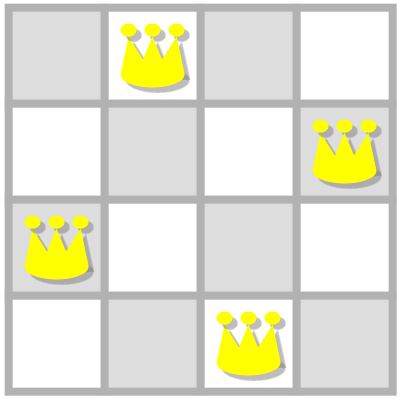
- The objective of this course is to study various fundamental, important, useful and interesting algorithms
- To learn solving complex problem
  - Learn how to organize the data and translate the real world problems into computer's internal representation
  - Analysis the problem and select the most suitable data structure and algorithm to design high speed and memory efficient programs

N queens problem

- To place *N* queens in a *N* x *N* chess board in the way such that the queens will not be captured by each other

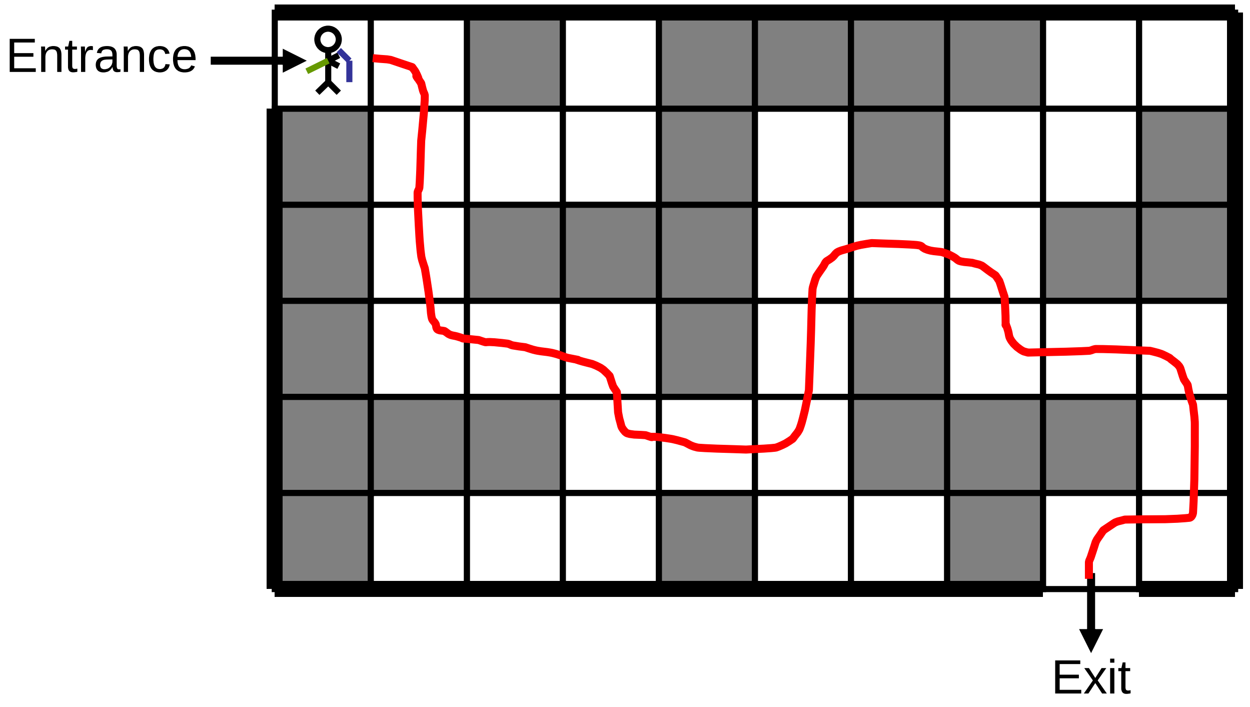

The queen can capture all other chess in the same row, column, or diagonals


Only can place 3 queens!


Place 4 queens!

This is one of the solutions

Maze Problem



Repetition Problem

- Determine if there are any duplicated numbers from a group of *n* numbers
- e.g. for the list of numbers:
  - {52, 61, 18, 70, 39, 48, 28, 57, 61, 39, 43}
  - 61 and 39 repeated twice
- Can you suggest an algorithm to find the duplicated numbers?

Sorting Problem

- How to sort a list of *n* numbers in ascending / descending order?
- *n* can be a very large number, e.g. over 1 million

Selection Problem

- Determine the *k*th largest number from a group of *n* numbers
- e.g. for the list of numbers:
  - {52, 61, 18, 70, 99, 48, 28, 57, 66, 39, 43}
  - The largest number is 99
  - The 2nd largest number is 70
  - …
  - The 11th largest number is 18
- Can you suggest an algorithm to determine the *k*th largest number?

Evaluating Math. Expressions

- How does a computer evaluate mathematical expressions?
  - 1 + 2 + 3
  - 1 + 2 x 3 //“2 x 3” should be calculated first
  - (1 + 2) x 3 //“1 + 2” should be calculated first
  - 1 x 2 + 3 x 4

Data Structures

- How to represent a fraction?
  - e.g. how to represent ½ in your program?
    - float x = 0.5; (floating pointing number)
  - e.g. how to represent 1/3 in your program?
    - float x = 0.333;
      - Not an exact number!
    - float y = 1.0 / 3;
      - *y* will be roughly equal to 0.33333333…
      - Still not an exact number!

Data Structures

- Declaring a small-size array but later find that the size is not enough

```
int a[3];
…
//later want more array elements!
```

- Declaring a large-size array but later find that much memory has been wasted

```
int a[100];
…
//just used several array elements only!
```

- Do we have dynamic growing/shrinking data structures?

Students are Expected …

- To have experience in C++ programming
- Some exposure to elementary algorithms on simple data structures such as arrays will be helpful

- Lecture: You build up the concepts
- Tutorial: You implement what you learned
  - Important! A process of transforming subtle and unshaped concepts to concrete and solid knowledge
  - Don’t memorize code. Algorithms are dynamic and alive!
- Home:
  - Skim the notes before you come to a lecture. Jot down what you don’t understand.
  - Review the notes after a lecture. Bring your questions to me. You are encouraged to share your problems/solutions in the course forum.

In-Class Exercise

- A **palindrome** is a word that spells the same backward or forward (e.g. *level*, *refer*). An **anagram** is a word formed from another by rearranging its letters (e.g. *listen* is an anagram of *silent*). Given a string of input, design an algorithm to check if the string is <u>an anagram of palindrome</u>. For example, consider the string "*aabbc*" which is an anagram of the palindrome "*abcba*".

- What data structure do you use in your algorithm?

Setting up your development environment

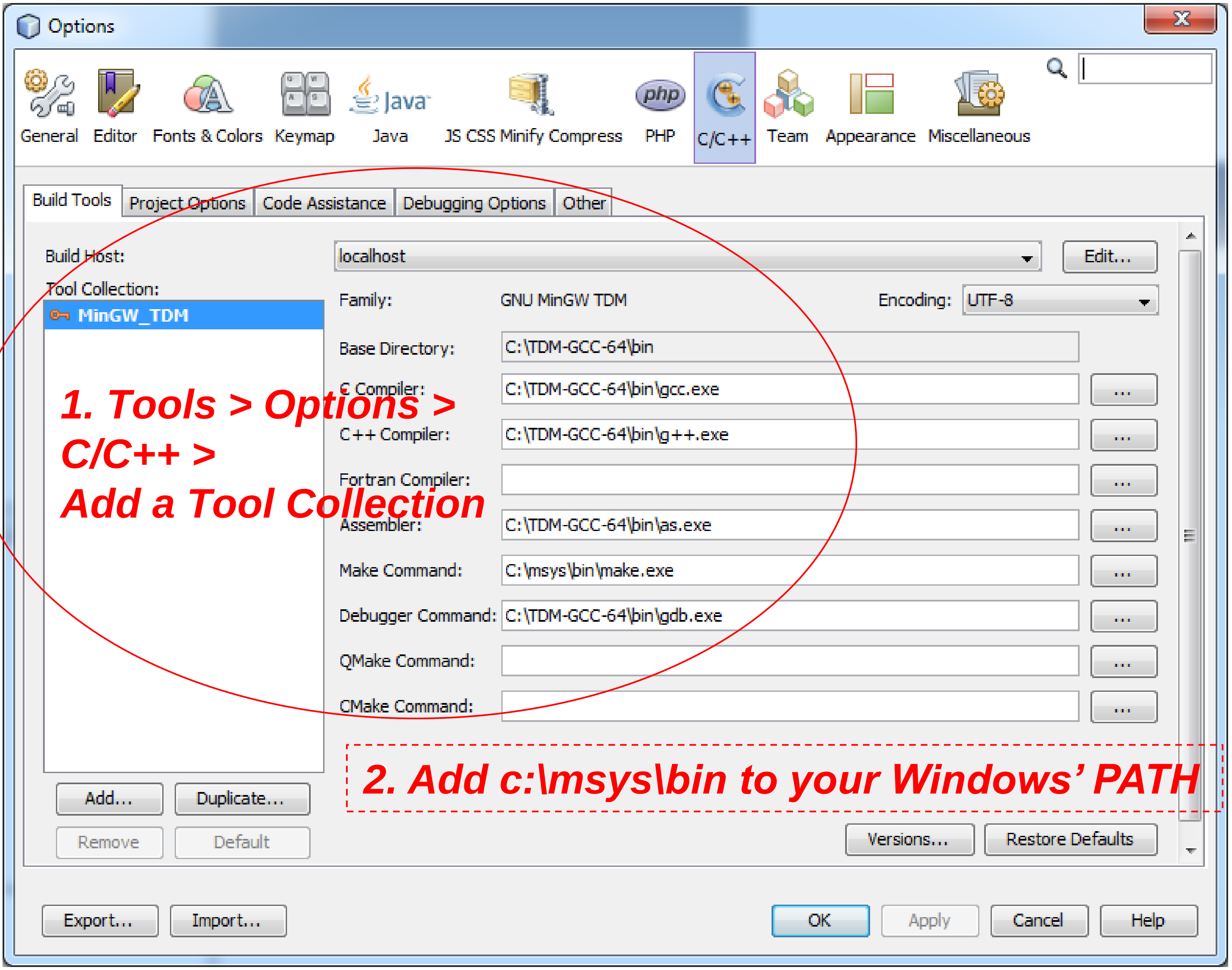
NETBEANS / GCC

Our Development Environment (Windows)

- Download and install/unzip the packages below:

- Netbeans with C/C++ Support
  - https://netbeans.org/downloads/
- GNU C/C++ Compiler for Windows
  - GCC 5.1.0 64-/32-bit
  - http://tdm-gcc.tdragon.net/download
- MSYS (Make Command)
  - https://sourceforge.net/projects/mingw-w64/files/External%20binary%20packages%20%28Win64%20hosted%29/MSYS%20%2832-bit%29/MSYS-20111123.zip/download
- **GCC is also available under Linux and MacOS but make sure your installed GCC version is same as stated above.*

Netbeans Configuration for C/C++



Build Host: localhost

Tool Collection: MinGW_TDM

Family: GNU MinGW TDM

Encoding: UTF-8

Base Directory: C:\TDM-GCC-64\bin

C Compiler: C:\TDM-GCC-64\bin\gcc.exe

C++ Compiler: C:\TDM-GCC-64\bin\g++.exe

Fortran Compiler:

Assembler: C:\TDM-GCC-64\bin\as.exe

Make Command: C:\msys\bin\make.exe

Debugger Command: C:\TDM-GCC-64\bin\gdb.exe

QMake Command:

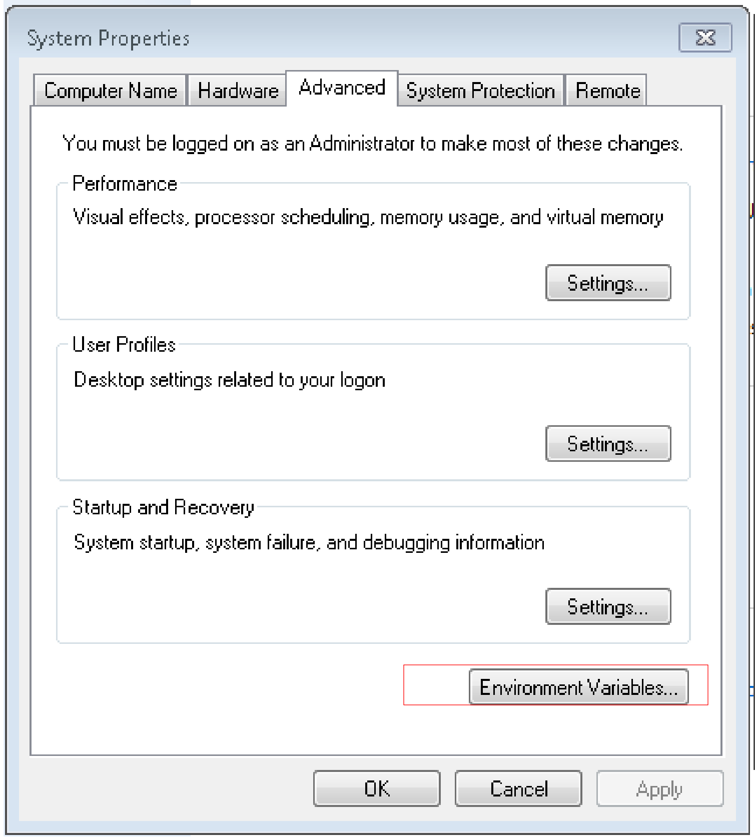
CMake Command:

1. Tools > Options > C/C++ > Add a Tool Collection

2. Add c:\msys\bin to your Windows' PATH

MSYS Configuration

- Go to Windows' "System Properties – Advanced"
- Click "Environment Variables"
- Add c:\msys\bin to your Windows' Path (assume your extracted it to C drive)


System Properties

Computer Name | Hardware | Advanced | System Protection | Remote

You must be logged on as an Administrator to make most of these changes.

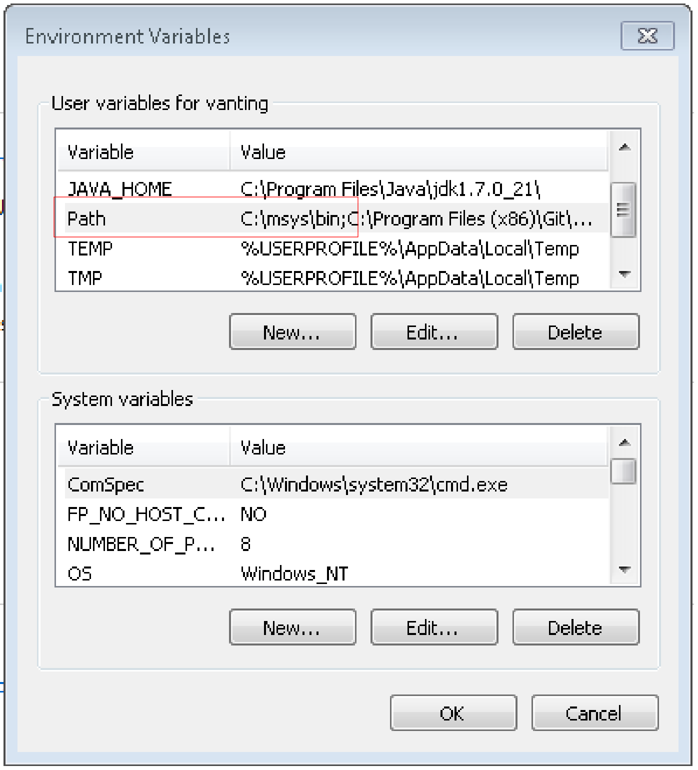
Performance
Visual effects, processor scheduling, memory usage, and virtual memory
Settings...

User Profiles
Desktop settings related to your logon
Settings...

Startup and Recovery
System startup, system failure, and debugging information
Settings...

Environment Variables...

OK | Cancel | Apply


Environment Variables

User variables for vanting

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk1.7.0_21\
Path	C:\msys\bin;C:\Program Files (x86)\Git\...
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

New... | Edit... | Delete

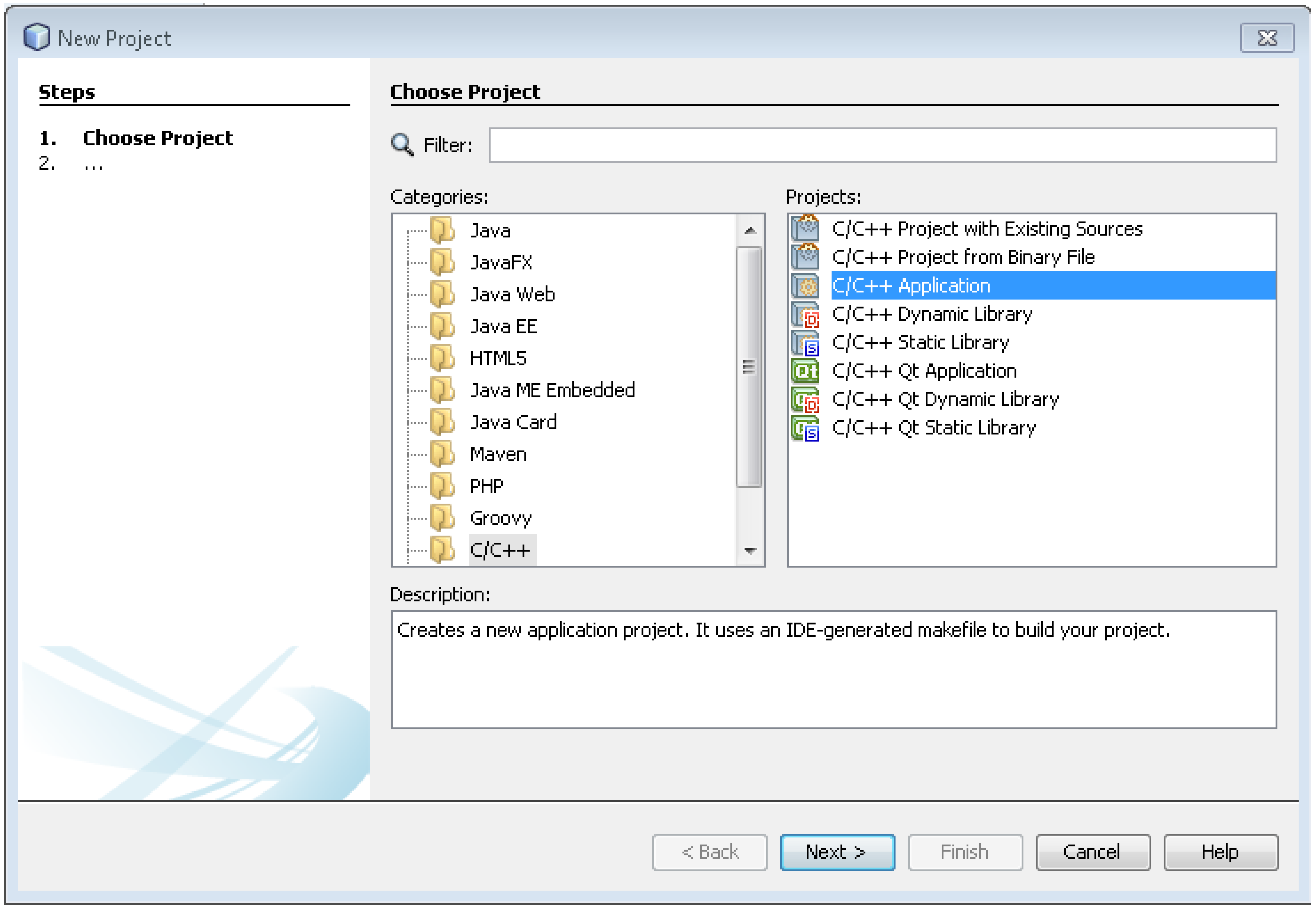
System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_P...	8
OS	Windows_NT

New... | Edit... | Delete

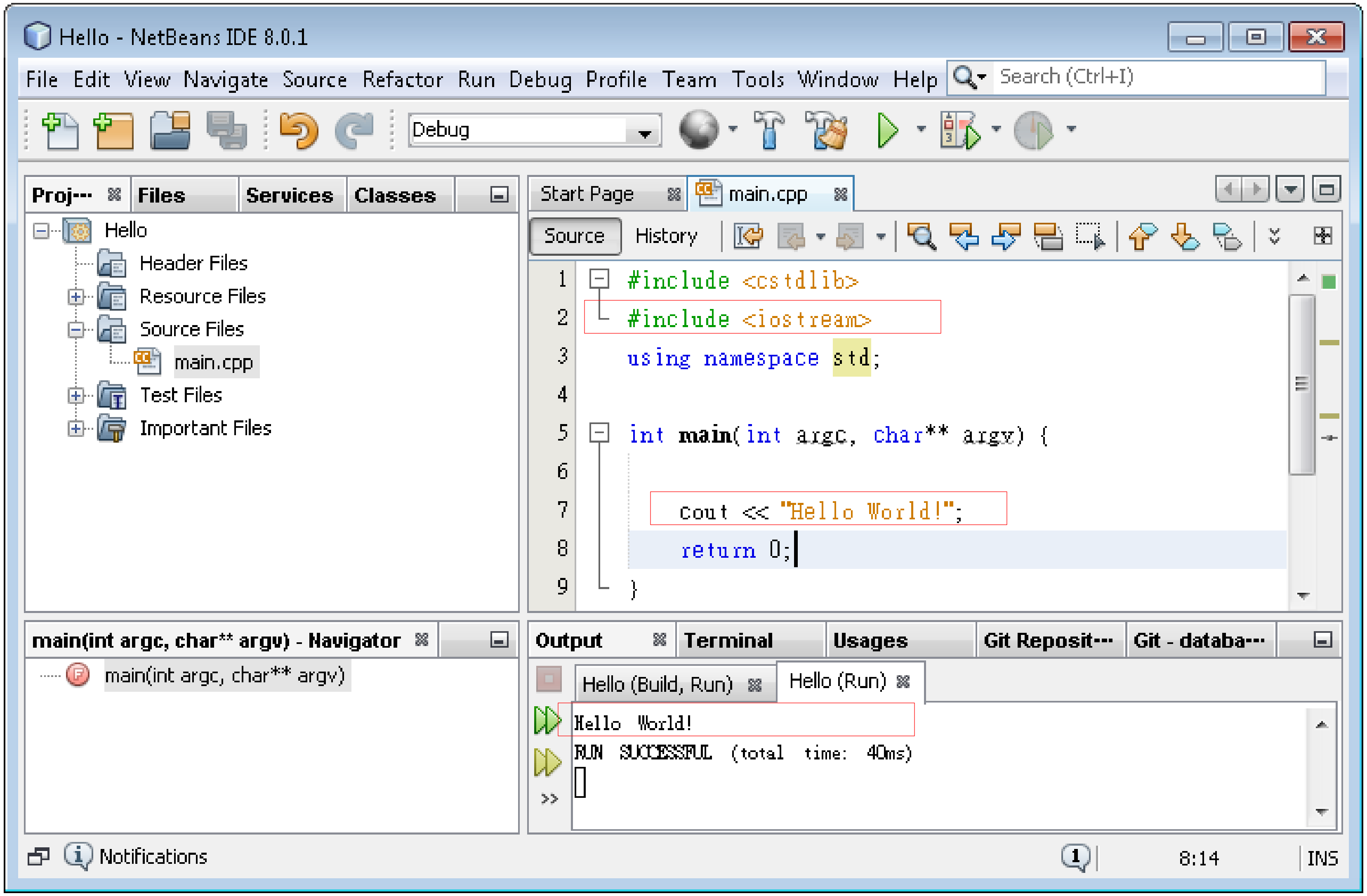
OK | Cancel

Create C++ Project



Creates a new application project. It uses an IDE-generated makefile to build your project.

Hello World IDE Test


Hello - NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Debug
Proj··· Files Services Classes
Hello
Header Files
Resource Files
Source Files
main.cpp
Test Files
Important Files
Start Page
main.cpp
Source History
main(int argc, char** argv) - Navigator
main(int argc, char** argv)
Output Terminal Usages Git Reposit··· Git - databa···
Hello (Build, Run)
Hello (Run)
Notifications
8:14
INS

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char** argv) {

    cout << "Hello World!";
    return 0;
}
```

```
Hello World!
RUN SUCCESSFUL (total time: 40ms)
```