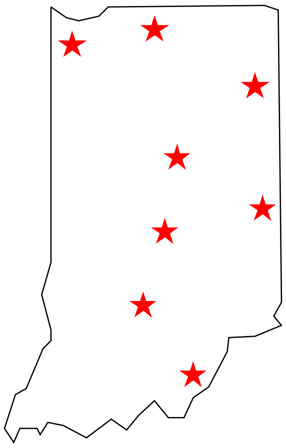
IUMMCC



Indiana University
Multi-campus Modular
Computing Curriculum

Introduction to Computing

Module # 1: Problem Solving With The STAIR Technique

Goals

- Students will learn a general problem solving strategy that can be applied to problems.
- Students will learn basic keyboard and mouse usage skills.
- Students will learn to identify key problem - solving tools in modern software applications.
- Students will apply problem solving strategies by learning how to play unfamiliar computer games.

Prerequisite Skills

- None

1 The Problem

Computers and related technology have become an undeniable part of our daily existence. It is almost impossible to live without interacting in some way with computing technology. The purpose of this course is to give you some tools you can use to begin mastering technology. We will not be able to make the impact of technology go away, but if we approach it carefully, we can decide whether our lives are controlled or enhanced by the computers that surround us. As the influx of technology grows at unprecedented rates, we quickly find that our accustomed ways of thinking may need refinement.

The universal[1] nature of computers makes them much more flexible than the kinds of machines we may be used to learning. Because of this flexibility, we cannot train ourselves to use a computer in the same way we teach ourselves to use a more mechanical machine, such as a car or a sewing machine. When you learn to use one of these traditional machines, you determine what it is

[1] A *universal* device is one capable of simulating the behavior of all other devices.

supposed to do, how to make it do that job, and what to do if it isn't working. Once you learn one car or sewing machine, the skill of driving or sewing is relatively easy to transfer to new cars or sewing machines. You can expect any car to do the same basic task, although the cockpit layout may be different, the transmission may require more skill, and so on. The same can be said of the sewing machine. You might need to learn a new way to install thread into the machine, but the process of sewing does not substantially change from machine to machine. In these largely mechanical single purpose machines, you have to learn only one skill, although that skill may be very complex in nature. You don't expect cars to start flying, and you don't expect a sewing machine to ever suddenly start acting like a microwave.

As we will see, computers are very different from these machines in at least one critical way. While a car or a sewing machine is designed primarily to do a single type of job, the computer is designed to do many very different kinds of jobs. The same computer could be used as a writing tool, to simulate the effects of a roller coaster, to play parlor games, to do complex mathematical calculations, and to retrieve stock market quotes from Tokyo. These things could conceivably happen at the same time in the same computer. The skills you would need to control a computer when it is helping you edit documents might be very different than the skills you use to get the stock quotes. If we approach learning a computer using the same learning model that has served us well since the beginning of the industrial revolution, we can never stop learning. It seems impossible to learn how to use a computer, because each time we have it do something else, we have to learn how to control it in that particular context. Some of the applications you will want to use a computer for in the next few years have not yet been invented!

How then, can we hope to teach a class to use a computer? Many educators have used the car/sewing machine model, treating each software application package as a new and distinct machine. Using this model, you might learn how to use a specific word processing package, say WordPerfect 6.0 for windows. You can learn how to use this system essentially the same way you learned how to drive or use a sewing machine. We can feed you a list of commands, take you step by step through a number of typical processes, and guide you carefully through your first few projects. This is a comfortable model of learning, and it works very well for teaching specific programs. If you need to learn exactly how to use WordPerfect 6.0 for Windows, and that's all you will ever use your computer for, such an approach would be ideal.

Unfortunately, the learning model outlined above has some serious weaknesses when applied to learning computers and technology. The first problem is the universality of computers. If we teach you word processing, you have learned something useful, but you really haven't learned much about computers. They can do so much more than process words. As an educated person, you will be expected to know many other ways to use computers. We would not be doing our job if we taught only one type of application.

Another problem is the pace of changing technology. It is no exaggeration to say that any software we install in the lab today will probably be out of date by the end of the semester. New releases of major programs come so frequently that memorizing commands and the layout of the program is a futile process. Things will change. You can't rely on the menu structure of a program, the control key combinations, or anything else to stay the same.

Anther weakness in the above model is the variety of programs that are available in any one category. If we decide to teach you WordPerfect in our classes, and you end up working in an office which uses only Word for Windows, you could be lost. The commands you have dutifully memorized when the computer is running one application could do nothing at all in a different application. It is conceivable that a command you use to save documents in one program could be used to delete documents in another. The potential for confusion is obvious.

The prospects for ever learning how to use and control a computer sound quite gloomy, but it is not nearly as bad as it sounds. If you reflect on your classmates, family, and friends, you can undoubtedly think of a person who can sit down at any computer, run a program she has never seen before, and produce amazing results. How do these people do it? Why is it that some people can run a never - before - seen program and make it do things for them? What is their secret?

After studying the way such people work, we have noticed some important themes:

- These people are (generally) not super-intelligent people who spend their whole lives with computers. Ordinary people can learn how to control computers with amazing flexibility and skill.
- Skilled computer users usually have a strong general knowledge about how computers work and what they can do. They know the main types of applications that are used on computers, and they know what features they can expect to find within an application.
- Skilled computer users have little fear of technology. They frequently use educated guesses to determine how things work, and they don't worry if their guesses are sometimes incorrect. Instead, they tend to learn from mistakes.
- The most striking characteristic of skilled computer users is the presence of some kind of organized problem-solving strategy. Skilled computer users are skilled problem solvers. They rely far more on their ability to solve new problems and find answers independently than on previous knowledge of the commands of an application program. They are able to encounter new programs and situations with little apprehension, because they are confident their problem-solving ability will help them overcome lack of knowledge about the new situation. The goal of this course is to help you become the type of skilled computer user we have described above. We will teach you a framework for problem solving with computers, and help you apply that framework to various types of application programs. We will introduce you to what computers can do as well as the types of problems you may be able to solve with specific software applications. You will also be given many opportunities to develop your problem-solving abilities by using real applications on real computers.

We will not focus on individual software packages, although we will certainly use them for examples and exercises. We will be far less interested in actual commands and methods for doing things, and more interested in the process you will use to learn how to do these things in whatever applications you use in the future. Our goal is to teach you the skills you will need to learn computing applications that have not even been imagined yet!

2 Our Most Powerful Tool - The STAIR Steps for Solving Problems

Problem-solving can be seen as more of an art than a science. Each person has different ways of approaching problems. Our intention is not to limit your problem-solving processes to our way of thinking, but to give you a framework you can fall back on when you are stuck. We have developed a method called STAIR which has proven to be a helpful problem-solving tool.

STAIR is an acronym. (Get used to it... Computer scientists LOVE acronyms!!) Each letter in the word STAIR represents a part of a general problem-solving strategy.

S - State the problem This step seems obvious enough, but it is often the one people skip. We frequently start solving problems before we really think carefully about what problem we are trying to solve. The result is confusion and wasted time. Take the time to describe carefully to yourself what you are trying to accomplish. Avoid the temptation to couch this problem description in technical terms. At this stage of the process, all your thoughts should be in ENGLISH (or, of course, whatever other human language you think most comfortably in). Some problems are best described with sketches or other tools (like flowcharts and data diagrams) which will be discussed later. Use whatever you need to make sure you have a firm understanding of what you want to accomplish. It's best to write this down on paper so the problem statement can be checked later if you get confused. If you don't know where you are trying to go, how will you know when you get there?

T - Tools for the job Once we know what kind of a job we want to accomplish, we can begin to explore which tools we might use to solve the job. We are using the term tool in a very broad sense. A tool might be a command, a button on a toolbar, a selection on a drop-down menu, a strategy, a program, or something else, depending on the kind of job we are trying to do and the context in which we are working. Knowing the capabilities of a certain program, and the kinds of things different types of computer applications can do are the main ways you add new tools to your toolbox of computing skills. As you gain experience, you will constantly be adding new tools of all kinds to your repertoire. Even the STAIR strategy itself can be considered a tool. Most of the time, there is more than one tool available to do a job. At this point, just list the available tools. Later on, you will decide which tool or tools to use. You might need to come back to this list if you chose an inappropriate tool the first time through the process.

A - Algorithm development An algorithm is a computer science term for a strategy or plan of action. Part of developing an algorithm is choosing an appropriate tool or set of tools from the previous step. The other component is determining how those tools will be used to solve the problem. As with the other steps, the algorithm can vary widely in the type and complexity of the strategy you will use. In simple problems, your algorithm may be as simple as saying "I'm going to try pressing this particular button on the menu bar." A more complex problem will probably require a more complex algorithm. As before, it is a good practice to write down your algorithm on paper.

I - Implementation of the algorithm It is interesting to note that none of the above steps require the use of a computer. The implementation step is the actual process of translating our human thought into something the computer can understand. In simple problems, our algorithm may be implementing a command or two. Implementation can mean "just do it" when we are dealing with these simple problems. Programmers think of the implementation step as translating the algorithm into some type of computer programming language. The process is the same regardless of the complexity of the problem.

R - Refinement We like to think if we learn a skill and prepare ourselves properly, we can solve a problem on the first attempt. Experience shows us this is not usually the case. It is normal for a computer user to attempt a solution several times before the problem is solved. A skilled problem solver will analyze what happened, review the other steps, and try again. Each unsuccessful attempt should bring you closer to an understanding of the problem and its solution. Refinement usually means going back and looking at the previous steps critically. Ask yourself if you really defined the problem properly. If so, have you used all the possible tools at your disposal? Are you sure there is not a tool available that you have overlooked? Did you choose the best algorithm for the job? Did you implement the solution properly? (You would be amazed at the number of computer errors that are the result of simple typing or spelling errors!) Again, you will find that practice will make you much more confident at this critical stage of the process.

3 Examples

As you can see, the STAIR process does provide some guidelines for thinking about a problem, but it is not a rigid technique. You will have to find your own variations that work well with the way you visualize and think. Many times you will find yourself not needing the entire process. The STAIR convention is a tool which may supplement your natural problem-solving processes when they are insufficient to solve your particular problem. It can also be used to help you organize your natural problem solving processes.

3.1 Example One: Storing the Boat

The STAIR process can be applied to situations that have nothing to do with computing. As an example, imagine the following situation: Tom came home one day to be confronted by his wife about the small sailboat that was filling space in the garage. She suggested he find a more efficient way to store the boat than leaving it where she normally parked her car. (Her suggestion was couched in somewhat stronger terms than this.!..) Being quite a good problem solver, he decided to apply the STAIR methodology to this problem.

Statement of the Problem Tom thinks to himself: "I need to get this boat out of the way. It needs to go someplace where it won't be in my wife's way, but with easy access when I want to go sailing. I don't want to spend a lot of money or effort on this problem, and I want a long term solution so I don't have to deal with this again".

Notice that Tom started with as simple a statement of the problem as possible. He also listed some of the relevant constraints. He didn't worry about what the exact solution might be yet. This may not be the finished statement of the problem. Tom may find he needs to come back and refine his problem statement a little more before he can finish the solution.

Tools Available Tom thinks: "This job might require actual physical hand tools, and I've got a pretty good set of the standard tools. Besides the obvious definition of the word tool, I might think of some of the main approaches to solving the problem as tools. I could hang the boat from the ceiling of the garage, attach it to the walls, put it in the yard, or rent some storage space."

Notice how flexible he was with the term 'tool.' This approach allows Tom to think of the IDEAS he will choose from, as well as the physical entities. In fact, the actual mechanical tools that will be used will be decided much later in the problem solving process. It's way too early to worry about whether he will use a hammer or a screwdriver.

Algorithm Tom thinks: "I'll try hanging the boat from the ceiling. This seems like the best approach, because the boat will be ugly in the yard, I can't afford rental space, and the walls are already full. I think the boat is light enough to be suspended from the ceiling safely with some ropes".

In this instance, the algorithm step involved choosing the best alternative from the tools step. At this point, he had made a decision about how he was going to tackle the problem. Notice that he also started thinking about which physical tools (in this case a rope) he would use to help solve the problem.

Implementation Tom says to himself: "When the time actually came to suspend the boat, I quickly realized I had more to think about. I measured the ceiling, found a perfect spot, and moved the boat to the floor directly under the spot. However, I didn't know how exactly to attach the boat to the ceiling. The ceiling has wooden rafters in it and obviously I need to attach some sort of hooks to the rafters. I also noticed the rope I planned to use was kind of slippery and might not hold a knot very well.

The implementation step so far has been unsuccessful in the obvious sense; the boat is still sitting on the garage floor. Rather than being discouraged by this, Tom will use the experience as an information gathering stage for the next step:

Refinement Tom thinks: "The ceiling idea is still a good one, but I don't have everything I need. I don't trust this rope, and I need some hooks or something. Digging through the toolbox, I found one bicycle hook that will work fine for the bow of the boat. I can attach it easily to a rafter, but I still need to hold up the back end of the boat. I noticed the straps I use to carry the boat on top of my car. They are sufficiently strong enough, but the ends of the straps are designed to fit in the rain gutter of my car. I need to find something that will allow me to attach the straps securely to the rafters. I went to the hardware store and found screen door handles! They will attach to the rafters and they are the perfect size to attach the straps to! I went back home, attached the handles, hoisted the boat up in the straps, and the problem is solved!"

The refinement step made some visits back to many of the other parts of the problem-solving process. Our handyman had to adjust his perception of the problem. Attaching the boat to the ceiling was a good idea, but Tom's first statement of the problem did not specify how he was going to do this. Finding a way to solve this part of the problem involved combining algorithm development and tool selection. The implementation happened in segments as he found more of the right ideas and tools to solve the problem. The process does not always flow in a set order, but because he used a systematic approach, he always knew how to think about solving the problem. Tom knew, for example, that his algorithm was good, but he would need to rethink his tool selection. Obviously, plenty of people solve this kind of problem without ever taking a computer science class. Tom may not even know that he was using an organized problem solving technique. The reason we begin with this problem is to illustrate that our problem solving-process is really just a somewhat formalized summary of the techniques we already recognize as problem solving techniques in many settings.

3.2 Example Two: Drawing a Map

Statement of the Problem Nancy has moved to a new town and her aunt wants to visit her. Nancy wants to use her new computer to draw a map from her aunt's home to her new house. She decided to use the computer so she can produce copies easily to send to other friends later.

Tools Available Nancy thinks carefully about the tools she needs. She knows her computer has a drawing program built-in, but she has never used the program. She does know how to get to the program and start it up.

Algorithm Nancy starts by drawing out the map with pencil and paper. Now she knows the image she wants to create. She will use lines to indicate streets, squares for specific landmarks, and text to label the streets and landmarks. When she has finished with the picture, she will save it to her disk in case she needs it again, and print it out to include in her next letter.

Implementation Nancy stares at the blank screen. She doesn't know what to do next. She knows that she has a number of new smaller problems:

- She needs to draw some rectangles.
- She needs to draw some lines.
- She needs to place some text on the picture.
- She needs to save the picture somehow.
- She needs to print the picture somehow.

Nancy has a dim recollection of the STAIR process from a computer class in her distant past. She wonders if she should apply that process to one of the smaller problems now. She decides to start with the rectangles:

- Statement of Problem: Nancy needs to draw a rectangle to represent her Aunt's house.
- Tools: She spots a button on the toolbar that has a picture of a rectangle! If she doesn't see it there, she knows she can cruise the menus for a likely looking command, or check the on-line help.
- Algorithm: She decides: "What the heck. I'll press it and see what happens."
- Implementation: She presses the button. The cursor on this screen changes, and when she drags the mouse on the screen, she is able to draw a rectangle. She practices a few times until she has the hang of it.

- Refinement: After she knows how to make a rectangle exactly how she wishes, she puts one where her aunt's house should be on the map.

Refinement Nancy repeats the process to learn the other tools she might need. Most of the things she needs relate directly to buttons on the toolbar. She does not see a button that might represent printing. She looks in the on-line help and finds that Print is an option in the File menu. She then looks at the menu and finds the command right there. She uses it and the picture shows up in her printer. Problem solved!

Notice that the problem solving process actually had to occur at two distinct levels in this project. First, Nancy had to think about the big picture of how to draw the map. As part of this larger problem, she discovered a number of smaller problems along the way. She applied the same kind of process to the smaller problems and was able to learn many new skills as part of solving the larger problem. The next time she needs to use this graphics program, she will already know how to use these particular tools without having to re-learn them. In fact, if she tries ANY graphics program, she will have a head start, because she already knows to expect some kind of line drawing tool, rectangle tool, and other tools that she learned to use in this program.

4 Laboratory Assignment

Today in lab you will use a special computer program designed to introduce you to the most basic computer skills. You will practice using the keyboard, editing text, locating special keys, using the mouse, using drop down menus, and accessing on-line help. After you finish the program, you will get to play some games. The lab instructor will NOT tell you how to play the games. It is up to you to figure out how the games work and how to play them. Use the STAIR technique to figure out how each game works. The lab instructor will demonstrate how to start the program.

Computers are universal machines - the same machine can do many different kinds of tasks.

Learning to use computers effectively requires problem solving ability.

The STAIR method is one way of organizing the problem solving process.

Statement of Problem: defining with words, sketches, or other means what you want to accomplish.

Tools: Concepts, Commands, Menu items, Toolbar buttons, programs, or ideas that can help you solve a problem.

Algorithm: A strategy for solving a problem.

Refinement: The process of studying an unsuccessful attempt at problem solving and looking back at earlier steps to approach a solution

5 Summary

The way we learn about using most machines is not adequate when we are trying to learn how to use a computer. Problem solving ability is a major factor in success with computers. The STAIR technique for problem solving is a good way to summarize the problem solving process. Problem solving is a fluid process. It may be necessary to apply the process to a large problem, to break that problem into many smaller problems, and apply the same process to the smaller problems.

5.1 Review Questions

1. Think of the way you already solve non-computing problems.

Q. 1 Do you have some type of systematic technique for solving problems?

(Answer on page 1-10.)

Q. 2 What do you do when you are stuck?

(Answer on page 1-10.)

Can you see some relationship between your current technique and the STAIR method?

2. Imagine that your friend calls you and tells you a skunk has gotten into his basement. Use the STAIR method to summarize how you will 'de-skunk' the basement.

3. Think of a problem you solved recently. Describe how you solved the problem in terms of the STAIR method. Write down how your thinking fit with the STAIR steps.

4. As you learn one of the games in today's lab, try to apply the STAIR technique to learning how to play the game. Write down the steps of the STAIR method, and how you used each part of the process to learn the game.

Answers to Questions

1. Possibly not, but that's why you are taking this course.

2. Scream, Cry, Gnash your teeth!