Debugging your way to success!

If there is one constant you can be assured of when stepping into the world of Computing Science (CS) and coding, it is that you will inevitably make mistakes along the way.

CS is awesome; whether you and your learners are delving into unplugged lessons, engaging with micro:bits, or fortunate enough to own robots you can program to drive, sing, tell stories and even dance!

From experience, one of the guaranteed benefits of engaging with CS and programming is the enthusiasm and willingness displayed by learners. However, another guarantee is that mistakes will be made, and the skill of debugging will be essential!

What is debugging, and why is it so significant?

Debugging is the process of identifying and resolving errors in code. Although this may seem straightforward, it requires learners to develop specialised skills. While debugging is closely tied to CS and programming, it not only aids in coding but also enhances crucial problem-solving abilities.

Debugging, or at least the bugs themselves, can sometimes carry negative connotations. Throughout all of the training we run via SSERC Digital, we discuss barriers to engaging with CS. One of the most common difficulties cited by teachers is a lack of confidence and uncertainty in overcoming issues when programming. The fear of not knowing how to support learners when their code encounters problems is a real concern and can contribute to disengagement.

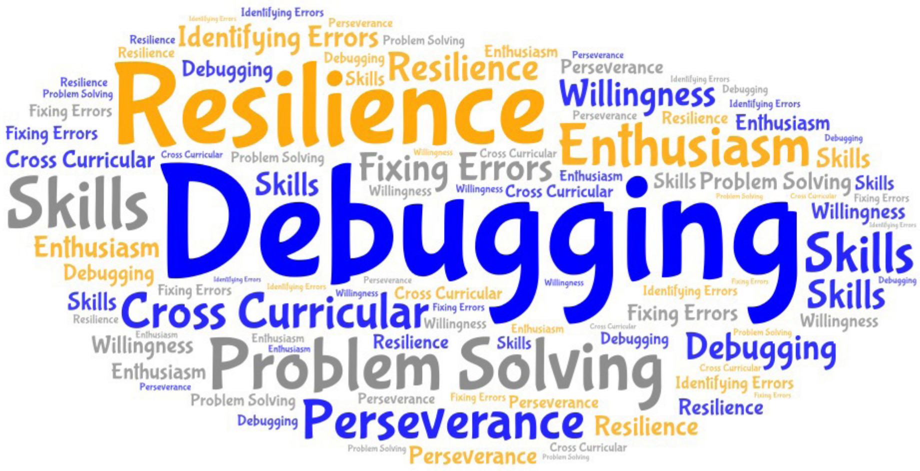
Despite the challenges, Computing Science offers a unique opportunity to enhance problem-solving skills, foster collaboration, and develop life skills. While some learners may struggle with debugging or problem-solving, the potential benefits of embracing Computing Science far outweigh these challenges.

Enthusiasm and willingness are vital to enhancing learning. From years of delegate feedback, as well as personal experience working with primary-aged learners, it is evident that CS is a powerful motivator for learners. Learners often strive to succeed with CS, demonstrating resilience and perseverance in the face of difficulties, allowing them opportunities to develop vital problem-solving skills that are transferable across the curriculum. In many of our showcase sessions, we have heard from educators commenting that learners who usually disengage from learning and are frustrated by mistakes were willing to try again and again in CS sessions, ultimately experiencing success.

Introducing debugging

When teaching CS, educators need to make a conscious effort to introduce specific concepts. This is one potential pitfall to be aware of when using resources such as Marty, Sphero Indi, VEX GO, and micro: bit, to name a few. Engaging with these resources alone does not lead to learners developing an understanding of algorithms, repetition, and selection; these concepts need to be taught alongside using the devices.

Debugging, on the other hand, will present itself more naturally, as it is an essential skill for progressing and succeeding with any CS resource or task. Engaging with CS inevitably >>



leads to mistakes. Therefore, it is essential to discuss debugging and how we need to be able to locate and fix bugs in our code to ensure our programs work correctly.

If you are introducing a CS resource to learners, the very first input is an excellent opportunity to make an error and discuss bugs with them. It's even better if the educator makes the mistake! Learners often enjoy fixing mistakes made by others, and debugging is an excellent way for them to engage with CS without being overwhelmed by the task of creating their own code from scratch. Instead, they get to explore pre-existing code and try to identify where it has gone wrong.

If you are looking for a few ideas of where to get started with your debugging journey, click on the picture below and work through some examples using BBC, Barefoot, micro:bit and others:

Not just the what, but the how!

No matter where you are on your CS journey, whether introducing your pupils to CS for the first time or you are well along your journey, debugging must be an essential part of every lesson.

It is vital to teach your learners how to debug and problem-solve rather than simply explaining what debugging is. We hear a lot of feedback about the lack of problem-solving skills our learners possess, but are we taking the time to teach the skill? As discussed, teaching coding is an ideal opportunity to do this.

Below are some examples of methods to use when developing debugging skills (Figure 1).

Across the curriculum

Having introduced the concept of debugging, it can be useful and fun to start using the terminology across the curriculum, taking the learners' enthusiasm for coding into other areas. Instead of learners being asked to correct their work in maths, they can debug it. Learners can be debugging their writing to check for any errors in their spelling, grammar and punctuation. This helps to soften the finality of getting something 'wrong', flipping that mindset to a search for a bug that first needs to be identified and then fixed. In our showcase sessions, we hear a lot of examples of how effective this is and how a change in pedagogical approach and use of terminology can have profound results.

Debug your way to success

It is often true that debugging a piece of code can take more time than writing the actual code. Even now, after using many CS resources for years, we find that debugging is still essential to any coding success. It is also true that the more complex your code, the more necessary it is for you to use your debugging skills. However, these skills can be developed from the very first engagements with CS, such as unplugged experiences using a codable robot to navigate a maze, and maintain their importance as you continue through to block coding and beyond. If you want your learners to be successful in their coding, then teaching, developing, and enhancing their debugging skills is essential, and it is made easier due to the enthusiasm, perseverance, and resilience when approaching and delivering your CS lessons. As an added bonus, the overall impact on learners' problem-solving skills will ultimately help them with their work across all learning.

CHECK & TEST

Encourage learners to check their code at regular intervals and run the code to test it.

This can be particularly useful with larger programs.

STEP-BY-STEP

Some coding platforms will allow you to run through your code step by step so you can easily identify where the bugs are. Even if the platform doesn't allow that, working through the code methodically is a great way to search for bugs.

USING DECOMPOSITION SKILLS

Sometimes debugging a large piece of code can be overwhelming, so encourage learners to use their decomposition skills to break large problems into more manageable chunks, and then debug from there.

PAIR PROGRAMMING

Working with a partner allows one of the pair to be responsible for checking the code and helping with debugging. As with writing, it can be hard to identify your own mistakes, whereas a fresh set of eyes often helps.

Figure 1