Slide Deck: Understanding Inheritance in Code

Slide 1: Welcome!

Title: Learning About Family and Blood Types in Code\ **Subtitle:** A Simple Guide for Beginners\ **By:** TA's @ Projectstake Academy

---

Slide 2: What's the Goal?

We want to see how a person's blood type is passed down from parents using code.\ We will use:

- Special data containers (called structs)
- Repeating steps (called recursion)
- Random choices

Slide 3: Important Ideas

- Each person has **2 letters** that make their blood type: A, B, or O
- Each person has **2 parents** who give 1 letter each
- We will use a struct (a small person-like box of info)
- The first people in the family tree get random letters

Slide 4: How the Code Works

1. `create_family()` – makes the family
2. `print_family()` – shows the family and blood types
3. `free_family()` – cleans up the memory
4. `random_allele()` – gives us a random blood letter

Slide 5: Making a Person

```c
typedef struct person {
    struct person *parents[2];
    char alleles[2];
} person;
```

Each person:

- Has two parent links
- Has two letters for blood type

Slide 6: Step 1 – Create a Person

```
person *new_person = malloc(sizeof(person));
if (new_person == NULL) exit(1);
```

🟢Make space in memory for a new person

Slide 7: Step 2 – Add Parents

If the person has parents, we create them:

```
person *parent0 = create_family(ggenerations  - 1);
person *parent1 = create_family(generations  - 1);
```

🔁This repeats for grandparents, great-grandparents, etc.

Slide 8: Step 3 – Pick Alleles from Parents

```
new_person->parents[0] = parent0;
new_person->parents[1] = parent1;

new_person->alleles[0] = parent0->alleles[random() % 2];
new_person->alleles[1] = parent1->alleles[random() % 2];
```

🔠Pick one letter from each parent

Slide 9: Step 4 – Oldest Generation

If this person has no parents (they're at the top):

```
new_person->parents[0] = NULL;
new_person->parents[1] = NULL;
```

```
new_person->alleles[0] = random_allele();
new_person->alleles[1] = random_allele();
```

☸️Give them two random letters

Slide 10: Cleaning Up (free_family)

We don’t want to waste memory! So we remove everyone when we’re done:

```
if (p == NULL) return;
free_family(p->parents[0]);
free_family(p->parents[1]);
free(p);
```

6 Clean up from bottom to top

Slide 11: Showing the Family Tree

We print each person and their blood type:

```
Child (Gen 0): AB
    Parent (Gen 1): AO
        Grandparent (Gen 2): OO
```

♋Indentation shows family levels

Slide 12: Picking a Random Blood Letter

```
int r = random() % 3;
return (r == 0) ? 'A' : (r == 1) ? 'B' : 'O';
```

🔝Picks A, B, or O randomly

Slide 13: Steps Recap

1. Make a new person

2. Build their parents (if any)
3. Pick letters from parents or randomly
4. Show the family
5. Clean up memory

Slide 14: What You Learned

- How families can be built with code
- How to use random choices
- Why cleaning memory is important
- How to show a family tree with code

Slide 15: Try It Yourself!

- Change the number of generations to 4
- Try tracking more traits (like eye color)
- Add code that shows the actual blood type (like AB or O)

Slide 16: The End!

‼️ Thanks for learning with us!\ Keep exploring CS50x with Us! 🆕
