Féidearthachtaí as Cuimse
Infinite Possibilities

Week 11 - Tutorial

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

Programming - Week 11

Overview

- malloc()
- calloc()
- realloc()

Dynamic Memory Allocation (DMA)

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

- There will be situations when you don't know how much memory is required when a program will run. In these situations, DMA is the best course to allocate memory while the program is running.
- DMA is provided in 2 ways:
  - The malloc() function
  - The calloc() function

malloc

- **malloc** (short for **memory allocation**)
- Standard library function used to allocate a block of memory dynamically at runtime
  - declared in the <stdlib.h> header file
- malloc() allocates a contiguous block of memory and returns a pointer to the start of the allocated block.

malloc syntax


```
void *malloc(size_t size);
```

Where:

- **size**: The number of bytes to allocate.
- **Return Value**: A pointer to the first byte of the allocated memory block. If the allocation fails, it returns NULL.

malloc – key points

- The memory returned by malloc is **uninitialized**, meaning it contains garbage values. If you need memory initialized to zero, use calloc.
- The function returns a void* pointer, which can be cast to any other pointer type as needed.

malloc example

Create a program to fulfil the following requirements:

- Create a guessing game to allow a user to guess a specific integer number.
- The number to guess will be hardcoded in the C program.
- The user can define how many guesses they can enter. **Malloc must be used** to define the allocated memory block to store the user number guesses.
- If the user guesses the number within their allocation display a message that they won and the guesses they entered.
- If the user doesn't guess correctly display a message that the game is over and list the number to guess and all the guesses they entered.

Program Design

Break the problem into smaller problems

1. Create a simple c program
2. Define the number to guess
3. Declare an int pointer
4. Get the number of guesses the user wants
5. Use malloc to define the size of the memory block
6. Create a loop to get the user to enter their guesses and store them in the memory block. If the guess is correct end the loop, else continue until the user runs out of guesses.
7. When the game is over show the user all their guesses.

Sample Solution

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL

C program_1.c > ...

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int *arr;
    int number_to_guess = 47;
    int number_of_guesses;
    int guess_number = 0;

    printf("Welcome to the guessing game\n");
    printf("How many numbe guesses would you like: \n");
    scanf("%d", &number_of_guesses);

    // Allocate memory for 'number_of_guesses' integers
    arr = (int *)malloc(number_of_guesses * sizeof(int));

    if (arr == NULL) {
        // Handle allocation failure
        printf("Memory allocation failed\n");
        return 1;
    }

    do {
        printf("\n\nEnter your %d guess:", guess_number + 1);
        scanf("%d", & *(arr + guess_number));

        if ( *(arr + guess_number) == number_to_guess ) {
            printf("You guessed correctly, you win....\n");
            guess_number++;
            break;
        } else {
            printf("Incorrect Guess");
            guess_number++;
        }
    } while(guess_number < number_of_guesses);

    printf("\n\n ****** Game over!!!! ******");

    printf("\n\nYou were assigned %d number_of_guesses\n", number_of_guesses);
    printf("Number to guess was %d. Your guesses: ", number_to_guess);
    for (int i = 0; i < guess_number; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    // Free the allocated memory
    free(arr);

    return 0;
}
```

calloc

- **calloc** (short for *contiguous allocation*) is a standard library function used to allocate memory dynamically.
- It is similar to malloc but has an important distinction: it initializes the allocated memory to **zero**.
- It is declared in the <stdlib.h> header file.

calloc syntax


```
void *calloc(size_t num, size_t size);
```

Where:

- **num**: Number of elements to allocate.
- **size**: Size of each element in bytes.
- **Return Value**: A pointer to the allocated memory block. If the allocation fails, it returns NULL.

Program 1 - Refactor

- Change Program_1.c to use calloc in place of malloc.

Program 1 - Refactor

- Change Program_1.c to use calloc in place of malloc.

```
13
14      // Allocate memory for 'number_of_guesses' integers
15      //arr = (int *)malloc(number_of_guesses * sizeof(int));
16      arr = (int *)calloc(number_of_guesses, sizeof(int));
17
```

realloc

- **realloc** (short for *reallocate*) is a standard library function used to resize a block of memory that has already been allocated dynamically (using malloc, calloc, or even realloc itself).
- declared in the <stdlib.h> header file.

realloc - syntax


```
void *realloc(void *ptr, size_t new_size);
```

Where:

- **ptr**: Pointer to the previously allocated memory block. It can be NULL.
- **new_size**: The new size (in bytes) of the memory block.
- **Return Value**: A pointer to the newly allocated memory block. If the allocation fails, it returns NULL, but the original block remains unchanged.

Program 1 - Refactor

- Update the example to cater for the following requirements:
- If the user gets their final guess wrong ask them if they would like 2 more guesses. If yes use realloc to update the size of the memory block.

Refactored solution

```
9       char choice;
```

```
29          if ( *(arr + guess_number) == number_to_guess ) {
30              printf("You guessed correctly, you win....\n");
31              guess_number++;
32              break;
33          } else {
34              printf("Incorrect Guess");
35              guess_number++;
36              if (guess_number == number_of_guesses) {
37                  printf("\n\nWould you like 2 more guesses (y or n): ");
38                  scanf(" %c", &choice);
39                  if (choice == 'y') {
40                      // Resize the memory block
41                      arr = (int *)realloc(arr, (number_of_guesses+2) * sizeof(int));
42                      number_of_guesses = number_of_guesses + 2;
43                  }
44              }
45          }
```

Mandatory Lab Question

- Show how to initialise two 3x4 arrays (2-Dimensional arrays with 3 rows and 4 columns in each) when they are declared. In your program, declare a 3rd 3x4 array. Multiply each corresponding element in the 1st and 2nd array and store this product in the corresponding element of the 3rd array. For example, array3[0][0] = array1[0][0] x array2[0][0], array3[0][1] = array1[0][1] x array2[0][1], etc...

Solution

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

C mandatory.c > main()

```c
#include <stdio.h>

int main() {
    // Initialize two 3x4 arrays
    int array1[3][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
    };

    int array2[3][4] = {
        {2, 4, 6, 8},
        {1, 3, 5, 7},
        {9, 11, 13, 15}
    };

    // Declare a 3rd 3x4 array to store the product
    int array3[3][4];

    // Calculate the element-wise product
    for (int i = 0; i < 3; i++) { // Loop through rows
        for (int j = 0; j < 4; j++) { // Loop through columns
            array3[i][j] = array1[i][j] * array2[i][j];
        }
    }

    // Display the resulting array3
    printf("The resulting 3x4 array after multiplication:\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 4; j++) {
            printf("%d ", array3[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Questions