Task 1E: Operations with Double Pointers using GDB

Objective:
To understand double pointers (pointer to pointer) in C and learn how data can be modified indirectly using a double pointer. This task also involves observing memory addresses and data modification using GDB.

1. Program Code

```
#include <stdio.h>

void modifyValue(int **pptr) {
        **pptr = 200;
}

int main() {
        int x = 100;
        int *ptr = &x;
        int **pptr = &ptr;

        printf("Before modification: x = %d\n", x);
        modifyValue(pptr);
        printf("After modification: x = %d\n", x);

        return 0;
}
```

2. Compilation Instructions

Compile the program with debugging symbols:

```
gcc -g double_pointer.c -o double_pointer
```

```
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1C$ cd ..
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11$ cd Task_1E
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1E$ ll
total 12
drwxrwxr-x 2 student student 4096 Jan  1 02:38 ./
drwxrwxr-x 7 student student 4096 Jan  1 02:38 ../
-rw-rw-r-- 1 student student  280 Jan  1 02:38 double_pointer.c
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1E$ gcc -g double_pointer.c -o double_pointer
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1E$ ./double_pointer
Before modification: x = 100
After modification: x = 200
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1E$
```

3. GDB Commands Used

break main
run
print x
print &x
print ptr
print &ptr
print pptr
print &pptr
step
print **pptr

4. GDB Output

```
(gdb) break main
Breakpoint 1 at 0x1191: file double_pointer.c, line 7.
(gdb) run
Starting program: /home/student/25SUB4508_LSP/25SUB4508_56133/ClassWork/day11/Task_1E/double_pointer
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at double_pointer.c:7
7       int main() {
(gdb) print x
$1 = 0
(gdb) print &x
$2 = (int *) 0x7fffffffdef4
(gdb) print ptr
$3 = (int *) 0x0
(gdb) print &ptr
$4 = (int **) 0x7fffffffdef8
(gdb) print pptr
$5 = (int **) 0x0
(gdb) print &pptr
$6 = (int ***) 0x7fffffffdf00
```

5. Observations & Explanation

1. Variable x stores an integer value and resides in the stack.
2. Pointer ptr stores the address of x.
3. Double pointer pptr stores the address of ptr.
4. Using **pptr allows indirect modification of x.
5. All variables are stored in stack memory but reference different levels of indirection.

6. Memory Relationship Diagram (Conceptual)

pptr --> ptr --> x --> value
(address) (100 → 200)

7. Conclusion

This task demonstrates how double pointers enable indirect data manipulation in C. They are essential when functions need to modify pointer values or dynamically allocated memory references.