

```
/*0/1 knapsack(not global)*/
```

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

```
int knapsack_0_1(int w[], int p[], int c, int n);
```

```
int main() {
```

```
    int i, n, c;
```

```
    printf("Enter the array size for both weight & profit:");
```

```
    scanf("%d", &n);
```

```
    int w[n], p[n];
```

```
    printf("\nEnter the weights and profits");
```

```
    for (i = 0; i < n; i++) {
```

```
        printf("\nWeight[%d]:", i + 1); //id starting from 0 but showing it's starting from 1
```

```
        scanf("%d", &w[i]);
```

```
        printf("Profit[%d]:", i + 1); //id starting from 0 but showing it's starting from 1
```

```
        scanf("%d", &p[i]);
```

```
    }
```

```
    printf("\nEnter the capacity:");
```

```
    scanf("%d", &c);
```

```
    int max = knapsack_0_1(w, p, c, n);
```

```
    printf("Max profit: %d\n", max);
```

```
    return 0;
```

```
}
```

```
int knapsack_0_1(int w[], int p[], int c, int n) {
```

```
    int i, j, ct=0; // Initialize the selected item count
```

```
    int ks[n + 1][c + 1];
```

```
    for (i = 0; i <= c; i++) {
```

```
        ks[0][i] = 0;
```

```
}
```

```
for (i = 0; i <= n; i++) {
```

```
    ks[i][0] = 0;
```

```
}
```

```
for (i = 1; i <= n; i++) { //row    /*for each cols
```

```
    for (j = 1; j <= c; j++) { //col    of each rows the loop is iterating*/
```

```
        if ((w[i - 1] <= j) && ((p[i - 1] + ks[i - 1][j - w[i - 1]]) > ks[i - 1][j])) {
```

```
            ks[i][j] = p[i - 1] + ks[i - 1][j - w[i - 1]];
```

```
        } else {
```

```
            ks[i][j] = ks[i - 1][j];
```

```
        }
```

```
    }
```

```
}
```

```
printf("The matrix is\n");
```

```
for (i = 0; i <= n; i++) {
```

```
    for (j = 0; j <= c; j++) {
```

```
        printf("%d ", ks[i][j]);
```

```
    }
```

```
    printf("\n");
```

```
}
```

```
// Backtrack to find the selected items
```

```
i = n;
```

```
j = c;
```

```
while (i > 0 && j > 0) {
```

```
    if (ks[i][j] != ks[i - 1][j]) {
```

```
        printf("Object %d selected (Weight: %d, Profit: %d)\n", i, w[i - 1], p[i - 1]);
```

```
        j -= w[i - 1];
```

```

        i--;

    } else {

        i--;

    }

}

return ks[n][c];

}

```

The screenshot shows a Windows terminal window with the following text:

```

C:\Users\HP\OneDrive\Desktop >
Enter the array size for both weight & profit:3
Enter the weights and profits
Weight[1]:1
Profit[1]:10
Weight[2]:2
Profit[2]:12
Weight[3]:4
Profit[3]:28
Enter the capacity:6
The matrix is
0 0 0 0 0 0 0
0 10 10 10 10 10 10
0 10 12 22 22 22 22
0 10 12 22 28 38 40
Object 3 selected (Weight: 4, Profit: 28)
Object 2 selected (Weight: 2, Profit: 12)
Max profit: 40

-----
Process exited after 21.35 seconds with return value 0
Press any key to continue . . .

```

The terminal window has a taskbar at the bottom showing the system clock as 13:29 on 06-12-2023, and the weather as 74°F Cloudy.