Călăuz Răzvan

PROIECT DE LECȚIE

Școala: Liceul Teoretic “Nicolae Bălcescu” Cluj-Napoca
Profesor: Călăuz Răzvan
Clasa: 10 MI 1
Profil/Specializare: Matematică-Informatică
Data: 27.03.2025
Disciplina: Informatică
Unitatea de învățare: Tipul de date struct
Titlul lecției: Tipul de date înregistrare
Tipul lecției: consolidarea și aprofundarea cunoștințelor
Competențe generale: Identificarea datelor care intervin într-o problemă și aplicarea algoritmilor fundamentali de prelucrare a acestora
Competențe specifice:

1. Uilizarea șirurilor de caractere și a structurilor de date neomogene în modelarea unor situații problemă;
2. Implementarea unor algoritmi de prelucrare a șirurilor de caractere și a structurilor neomogene.

Obiective operaționale: La finalul lecției, elevul va fi capabil să:

- Explice importanța tipului de date înregistrare;
- Declare și inițializeze înregistrări în limbajul C++;
- Acceseze și folosească înregistrările într-o problemă practică;
- Implementeze operațiile de adăugare, citire și scriere din fișier pe o înregistrare.

Desfășurarea lecției:

Evenimentele lecţiei	**Activitatea din lecţie**	**Strategia didactică și evaluarea**
Captarea atenției	Discuție deschisă cu elevii și răspunderea la întrebări.	Conversația
Reactualizarea cunoștințelor anterior însușite	Punerea unor întrebări recapitulative: Pentru ce și în ce situații folosim înregistrări?	Evaluarea frontală
Informarea elevilor asupra obiectivelor urmărite	Titlul lecției: Gestionarea datelor folosind înregistrări	Conversația
Dirijarea învățării	**Activitate practică:** Într-un joc, avem **4 personaje** (un jucător și 3 inamici). Fiecare personaj are următoarele atribute: • Nume (maxim 50 de caractere) • Hp (număr întreg) • Atac (număr întreg reprezentând puterea de atac) Cerințe: 1. Să se creeze o structură numită **Personaj**, care `struct Personaj { char nume[51]; int hp; int atac; };` conține nume, hp și atac. 2. Să se creeze o funcție care citește din fișierul “input.txt” informațiile despre jucător, apoi informațiile despre fiecare inamic. Inamicii vor fi	Explicația, evaluarea frontală, evaluarea practică

memorați într-un vector de Personaje denumit **inamici**.

3. Să se citească, de la tastatură, un număr întreg R reprezentând **numărul de runde**.

```
void citestePersonaj(Personaj &jucator, Personaj inamici[]) {
    ifstream fin( s: "input.txt");

    fin >> jucator.nume >> jucator.hp >> jucator.atac;

    for (int i = 0; i < 3; i ++) {
        fin >> inamici[i].nume >> inamici[i].hp >> inamici[i].atac;
    }

    fin.close();
}
```

4. Să se creeze o funcție care, pentru fiecare rundă:

- Jucătorul alege pe cine atacă, introducând un număr între 1 și 3 (indexul inamicului) (atacul reprezintă scăderea valorii câmpului “hp” al inamicului de pe indexul dat de utilizator cu valoarea câmpului “atac” al jucătorului);
    - Dacă inamicul este deja eliminat (hp <= 0), atacul jucătorului este pierdut și se afișează mesajul “Atac ratat! Inamicul este deja eliminat!”;
    - Dacă inamicul nu este deja eliminat (hp > 0), se efectuează atacul și se afișează mesajul “[nume_jucator] ataca [nume_inamic] cu [atac_jucator] damage!”
- După atacul jucătorului, toți inamicii **rămași** (hp > 0) atacă jucătorul o dată fiecare;
- Dacă jucătorul ajunge la hp <= 0, lupta se încheie imediat;
- Dacă toți inamicii sunt eliminați (hp-ul fiecărui inamic <= 0), lupta se încheie imediat.

```
void executaRunda(Personaj jucator, Personaj inamici[]) {
    int indexInamic;
    cout << "Alege inamicul: 1 - Goblin, 2 - Dragon, 3 - Orc -> ";
    cin >> indexInamic;

    indexInamic --;

    cout << "Runda 1: " << endl;

    // Jucatorul ataca
    if (inamici[indexInamic].hp <= 0) {
        cout << "Atac ratat! Inamicul este deja eliminat!";
    } else {
        inamici[indexInamic].hp -= jucator.atac;
        cout << jucator.nume << " ataca " << inamici[indexInamic].nume <<
            " cu " << jucator.atac << " damage!\n";
    }

    // Inamicii ataca
    for (int i = 0; i < 3; i ++) {
        if (inamici[i].hp > 0) {
            jucator.hp -= inamici[i].atac;
        }
    }
}
```

5. Să se creeze o funcție care verifică dacă jocul s-a sfârșit:

<table>
<tr>
<td></td>
<td>

După fiecare rundă, afișează punctele de viață (hp) ale jucătorului și ale inamicilor;
Dacă jucătorul a fost eliminat (hp <= 0), se afișează mesajul “Jucatorul a fost eliminat!” și se încheie jocul;
Dacă toți inamicii au fost eliminați, se afișează mesajul “Toti inamicii au fost invinsi!” și se încheie jocul;


```
bool verificaFinalJoc(Personaj jucator, Personaj inamici[]) {
    // Verificam daca jucatorul a fost eliminat (hp <= 0)
    if (jucator.hp <= 0) {
        cout << "Jucatorul a fost eliminat!\n";
        return true;
    }

    // Verificam daca toti inamicii au fost eliminati
    int nrInamiciEliminati = 0;
    for (int i = 0; i < 3; i ++) {
        if (inamici[i].hp <= 0) {
            nrInamiciEliminati ++;
        }
    }
    if (nrInamiciEliminati == 3) {
        cout << "Toti inamicii au fost invinsi!";
        return true;
    }

    // Lupta continua
    return false;
}
```

6. Să se creeze o funcție care afișează statusul jocului (hp-ul fiecărui personaj).

```
void afiseazaStatus(Personaj jucator, Personaj inamici[]) {
    cout << jucator.nume << " HP: " << jucator.hp << endl;
    for (int i = 0; i < 3; i ++) {
        cout << inamici[i].nume << " HP: " << inamici[i].hp << endl;
    }
    cout << "=========================";
}
```

Bonus: Dacă se termină toate rundele și există supraviețuitori de ambele părți: “Lupta s-a incheiat fara un castigator!”.
</td>
<td></td>
</tr>
<tr>
<td>Asigurarea conexiunii inverse</td>
<td>Ce funcționalități am putea adăuga acestei probleme/proiect pentru a fi mai interesant?</td>
<td>Calculatorul, învățarea prin descoperire, problematizarea, dezbaterea</td>
</tr>
<tr>
<td>Asigurarea reținerii</td>
<td>Terminați proiectul și adăugați o funcționalitate nouă, diferită de celelalte, care să includă utilizarea Jucătorului și cel puțin a unui Inamic.</td>
<td>Tema pentru acasă, evaluare independentă</td>
</tr>
</table>