# CONCEPTE DE PROGRAMARE

**1. Cunoaşterea cererii de programe.**

Atunci când persoanele individuale sau firmele au nevoie de un program, există trei modalități de a-l obține:

- cumpărarea unui program deja scris. În acest fel programul poate fi obținut rapid şi nu costă mult. Există însă posibilitatea ca programul să nu execute exact ceea ce trebuie.
- Cumpărarea unui program şi modificarea sa astfel încât versiunea personalizată să facă exact ceea ce este necesar utilizatorului. Astfel programul poate fi transformat destul de rapid într-un program utilizabil, însă nu este totdeauna posibil să se modifice un program de pe piață.
- Scrierea unui program propriu. Astfel, programul face exact ceea ce doreşte utilizatorul însă această opțiune necesită mult timp şi costurile sunt foarte mari.

Majoritatea utilizatorilor de microcalculatoare se opresc la prima opțiune, deoarece programele sunt relativ necostisitoare, comparativ cu puterea lor. Deoarece firmele, cum ar fi Intuit, Microsoft, Borland şi Symantec, vând în cantitate foarte mare o versiune a unui program, pot să mențină preturi relativ scăzute. Utilizatorii individuali de microcalculatoare pur şi simplu nu dispun de resursele necesare pentru a scrie fiecare program de care au nevoie.

Pe de altă parte, firmele nu aleg întotdeauna prima opțiune. Firmele investesc mulți ani în realizarea acelor produse şi servicii care să le deosebească de alte firme. Atunci când o întreprindere se decide să-şi computerizeze oricare dintre evidențele sale, este de importanță vitală ca programele să reflecte exact modul de lucru utilizat deja în cadrul acesta. Firma nu trebuie să-şi modifice modul în care face afaceri, numai pentru a putea utiliza programele disponibile în magazine sau în livrările de software prin poştă. Programele de pe piață trebuie să fie generice, astfel încât producătorii acestora să le poată vinde mai multor cumpărători.

Cea de-a doua opțiune - de a cumpăra un program şi a-l personaliza - pare să fie cea mai bună, dar este aleasă cel mai rar. Dacă companiile ar putea cumpăra un program deja scris, ar trebui să aibă un cadru de lucru în care să-l adapteze rapid necesităților proprii. Problema este că programele sunt rareori vândute; în schimb, este vândută *licența* lor. Atunci când cumperi un program, nu devii proprietarul acestuia, ci capeți doar dreptul de a-l utiliza. Din punct de vedere legal, nu ai dreptul de a-l modifica, vinde sau copia (exceptând scopul salvărilor de siguranță). Aici sunt implicate nu numai aspecte legislative, ci uneori modificarea programelor nu este posibilă din punct de vedere fizic. Un program o dată scris, este transformat într-un format comprimată, pe care programatorii nu o mai pot modifica.

Rezultă că, deşi scrierea programelor de la zero este costisitoare şi necesită un timp îndelungat, majoritatea firmelor preferă să procedeze astfel, păstrând vaste departamente de programare pentru rezolvarea acestor chestiuni. Se poate întâmpla ca o firmă să plătească timp de un an întreg câțiva membri din echipa de prelucrare a datelor, pentru ca aceştia să scrie un program foarte asemănător - dar nu identic - cu unul pe care l-ar putea cumpăra de pe piață. În ciuda costului şi eforturilor implicate, pentru firmă merită să nu fie silită să se conformeze constrângerilor unui program cumpărat de la altcineva. O dată scris, programul corespunde modului în care se lucrează în firma respectivă.

Unele firme au descoperit că pot vinde programele realizate altor firme, ce se ocupă de afaceri similare, recuperând astfel o parte din costurile suportate. Atunci când scrieți programe pentru firma dumneavoastră sau pentru persoane individuale, rețineți că există posibilitatea vânzării ulterioare a acestor programe.

Adeseori, firmele măsoară timpul necesar pentru scrierea programelor în *ani-persoană.* Dacă sunt necesari doi ani-persoană pentru a scrie un program, se estimează că doi oameni ar putea să-l scrie într-un singur an sau o persoană în doi ani. Un proiect de 20 ani-persoană ar necesita 20 de persoane pe durata unui an, sau o persoană pe durata a 20 de ani, sau 10 persoane pe durata a doi ani şi aşa mai departe. Acest mod de apreciere este doar estimativ, dar oferă celor din management o estimare a modului în care trebuie să aloce oameni şi timp pentru proiectele de programare.

Dacă veți deveni un programator ce va executa lucrări pe bază de contract, estimarea în ani-persoană va constitui un instrument important pentru stabilirea prețului serviciilor dumneavoastră. Ați putea da clientului dumneavoastră o estimare a prețului în ani-persoană (sau, pentru proiecte mai mici, v-ați putea estima munca în luni-persoană sau săptămâni-persoană). Dacă veți angaja programatori care să vă ajute să terminați programul, veți putea încheia mai devreme şi totuşi veți putea să plătiți în mod corect munca fiecăruia, datorită faptului că ați calculat prețul în ani-persoană şi nu în funcție de timpul calendaristic.

De ce nu sunt deja scrise toate programele necesare vreodată? Intrați în orice magazin de software şi veți vedea sute de programe de vânzare. Există programe pentru orice: procesare de text, contabilitate, desen, jocuri, proiectare de locuințe, conectare on-line şi planificarea itinerariilor excursiilor. S-ar părea că aveți la îndemână orice program doriți. Deoarece calculatoarele au apărut de 50 de ani, se poate crede că toată lumea a cam terminat cu programarea necesară pentru mult timp de acum încolo.

Dacă toate programele necesare ar fi deja scrise, nu ați vedea lungile liste de afişe de tipul „Se caută programator" din ziarele actuale. Realitatea este că lumea se schimbă în fiecare zi, iar oamenii şi întreprinderile trebuie să se schimbe o dată cu ea. Programele scrise acum zece ani pur şi simplu nu mai corespund uzanțelor actuale. Pe lângă aceasta, au fost scrise pentru calculatoare mult mai lente şi mult mai limitate decât cele din zilele noastre. Pe măsură ce se fac progrese în ceea ce priveşte componentele hardware, programele trebuie să țină pasul cu acestea.

Astăzi, mai mult decât oricând, există o cerere extraordinară de programatori. Pe măsură ce calculatoarele devin mai uşor de utilizat, unele persoane sunt de părere că programatorii vor deveni relicve ale trecutului. Ceea ce nu iau însă aceştia în considerare este necesitatea de programatori de primă clasă, care să realizeze aceste programe uşor de utilizat.

**2. Definiții ale programelor**

Dacă v-ați orientat vreodată într-un teritoriu necunoscut după hartă, atunci ştiți cam ce înseamnă pentru calculator urmărirea instrucțiunilor dintr-un program. Având numai harta, sunteți ca orb atunci când vă mutați dintr-un loc într-altul şi o luați la stânga sau la dreapta, până când ajungeți la destinație sau până când constatați că, undeva pe parcurs, ați luat-o într-o direcție greşită. Calculatorul este o maşină oarbă, care aşteaptă de la dumneavoastră instrucțiuni. Atunci când i le dați, calculatorul îndeplineşte instrucțiunile, fără a ghici care sunt dorințele dumneavoastră aflate în spatele acestora. Dacă-i cereți unui PC să facă ceva incorect, va face tot ce poate ca să îndeplinească sarcina.

**Un program este o listă de instrucțiuni detaliate, pe care le execută calculatorul.**

Termenul de *„detaliat",* din definiția de mai sus, este vital pentru a face calculatorul să urmeze ordinele dumneavoastră. De fapt, sarcina unui programator nu este dificilă; dificilă este fragmentarea lucrării unui calculator în paşi simpli şi detaliați, care nu se bazează pe nici o presupunere.

Pentru a vă face o idee despre modul de gândire din programare, imaginați-vă cum ar trebui să descrieți cuiva din trecut pornirea unei maşini. Să presupunem că la uşa dumneavoastră apare un cowboy din vestul sălbatic, complet zăpăcit de priveliştea din jurul lui. După ce va fi trecut de şocul viitorului, el va dori să se adapteze noii lumi. Înainte de a învăța să conducă maşina, el va trebui să învețe să o pornească. Atunci când

va şti bine să facă aceasta, îl veți învăța să conducă. Spre deosebire de un puşti de 16 ani, el nu a crescut văzând adulții pornind maşini, aşa că trebuie să stăpânească cu adevărat acest proces, înainte de a putea merge mai departe. Fiind un programator foarte ocupat, îi lăsați următorul set de instrucțiuni agățat de cheia maşinii:

1. Foloseşte această cheie.
2. Porneşte maşina.

Cât de departe va ajunge el? Nu prea departe. I-ați dat instrucțiunile corecte de a porni maşina, dar ați presupus că ştie prea mult. Trebuie să vă amintiți că el nu ştie nimic despre aceste ciudățenii numite automobile şi că se bazează pe dumneavoastră ca să-i dați instrucțiuni pe care le poate înțelege, în loc de a presupune atât de mult, poate că acestea ar fi nişte instrucțiuni mai bune:

1. Cheia maşinii este anexată acestui bilet, îți trebuie ca să porneşti maşina.
2. Cu cheia în mână, mergi la uşa maşinii care se află cel mai aproape de uşa din față a casei noastre.
3. Sub mânerul negru al uşii, vei vedea o parte metalică rotundă, de dimensiunile unui ban, în care poți să introduci cheia (cu partea neregulată în jos).
4. După ce ai introdus cheia în gaură cât de mult se poate, răsuceşte-o spre dreapta până când vei auzi un clic.
5. Răsuceşte cheia înapoi spre stânga, până când ajunge în aceeaşi poziție în care ai băgat-o, apoi scoate-o.
6. Deschide uşa şi intră în maşină. Ai grijă să te aşezi în fața roții (denumite *volan),* aflate în partea stângă a locurilor din față.
7. Închide uşa.
8. În partea dreaptă a coloanei de susținere a volanului, vei vedea o scobitură în care poți pune cheia.

Lista de opt instrucțiuni este foarte detaliată şi cowboyul nici măcar n-a apucat încă să pornească maşina. Va mai trebui să descrieți pedala de accelerație pe care trebuie s-o apese atunci când răsuceşte cheia (în direcția corectă, se înțelege), apoi desigur că nu veți dori să vă bazați pe faptul că el va *opri* maşina atunci când va fi terminat de exersat, aşa că va trebui să-i dați şi aceste instrucțiuni.

În cazul în care credeți că această analogie cu pornitul maşinii a mers cam prea departe, gândiți-vă la ceea ce ar trebui să faceți pentru a-i putea spune unei piese de echipament electronic, care nu gândeşte - calculatorul - să efectueze statul de plată pentru firma dumneavoastră. Un program de stat de plată constă doar în următorii paşi:

1. Obținerea datelor pentru statul de plată.
2. Calcularea salariilor şi impozitelor.
3. Tipărirea cecurilor.

Pentru calculator, acestor instrucțiuni le lipsesc mii de detalii pe care dumneavoastră le considerați ca subînțelese. Detalierea instrucțiunilor programului este cea care produce exasperarea şi frustrarea ocazională în programare. A programa calculatoarele nu este dificil, dar a fragmenta problemele din realitate într-o mulțime de paşi detaliați - pe care-i poate înțelege calculatorul - este greu.

Un program obişnuit de stat de plată poate conține 20.000 de linii de instrucțiuni sau chiar mai mult. Totuşi, aceasta nu trebuie să vă descurajeze. Proiectele mari de programare din majoritatea firmelor sunt scrise de echipe de programatori; veți beneficia din plin de ajutor, dacă veți scrie vreodată astfel de programe pentru a trăi. Totodată, noile tehnici şi medii de programare utilizate în limbajele de programare actuale fac ca programarea - chiar pentru programatorul individual, ce lucrează de unul singur - să fie mult mai uşoară decât a fost vreodată în trecut.

1. Există multe instrumente de proiectare ce vă ajută să fragmentați problemele cuprinzătoare în componente detaliate, care se traduc în elemente de programare. Prin urmare, în loc să vă lăsați decepționat de aceste descrieri aparent înfricoşătoare ale detaliilor programelor, păstrați-vă încrederea pentru că ajutorul soseşte în curând. Totodată, țineți cont de aceasta dacă programarea ar fi fost într-adevăr dificilă, progresele din domeniul calculatoarelor din ultimii 50 de ani nu ar fi avut loc în nici un caz.

**3. Programele sunt instrucțiuni salvate**

Partea frumoasă în ceea ce priveşte programele pe care le scrieți este că pot fi salvate pe disc, atunci când sunt finalizate. Programele scrise se stochează în fişiere pe disc, tot aşa cum se procedează şi cu documentele realizate cu ajutorul procesoarelor de text. Un program este pentru calculator ca o rețetă pentru un bucătar. Atunci când un bucătar vrea să gătească o anumită mâncare, găseşte rețeta corespunzătoare şi urmează instrucțiunile acesteia. Când cineva vrea să execute un program, instruieşte calculatorul să încarce programul de pe disc în memorie şi apoi să ruleze instrucțiunile din acesta, în lecția precedentă a fost prezentată o listă detaliată de etape prin care trece calculatorul pentru a executa un singur program.

Memoria internă a calculatorului este de o importanță vitală pentru execuția programului. Unitatea CPU nu poate executa instrucțiunile unui program direct de pe disc. Tot aşa cum dumneavoastră nu puteți şti ce este într-o carte aflată pe masă până când nu citiți şi aveți în memorie conținutul acesteia (utilizând propria unitate CPU - mintea dumneavoastră), nici unitatea CPU din calculator nu poate prelucra instrucțiunile unui program până când nu-l încarcă de pe disc în memoria principală, în figura 1 este prezentat procesul de încărcare a unui program de pe discul calculatorului (sau o unitate de stocare asemănătoare discului, cum ar fi un CD-ROM) în memorie. Aşa cum se arată în figură, unitatea CPU are acces direct la memorie, dar nu are acces la unitatea de disc. Discul reprezintă stocarea pe termen lung, iar memoria reprezintă stocarea pe termen scurt, unde rezidă temporar programele în timp ce sunt executate de către unitatea CPU.

Rețineți diferența dintre program şi rezultatul său. Programul este un set de instrucțiuni, iar datele de *ieşire* sunt rezultatul acestor instrucțiuni. „Data de ieşire" a unei rețete este felul de mâncare terminat, iar datele de ieşire ale programului reprezintă rezultatul tipărit, o dată rulate instrucțiunile.

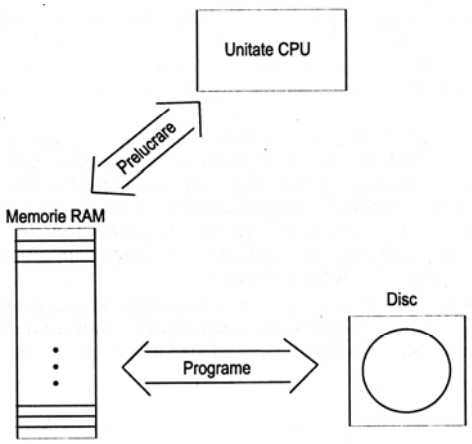


FIGURA 1

Un program trebuie să se afle în memorie (minte) ca unitatea CPU să-i poată executa instrucțiunile.

Poate că un exemplu ar ajuta la clarificarea a ceea ce înseamnă pentru un programator folosirea programului său de către utilizator. Dacă utilizați un procesor de text, probabil că executați o serie de paşi similari cu aceştia:

1. Încărcați programul de procesare de text de pe disc în memoria principală a calculatorului. Atunci când selectați din meniu denumirea programului de procesare de texte sau pictograma acestuia din Windows, dumneavoastră îi dați calculatorului instrucțiunea de a căuta pe unitatea de disc programul şi de a-l încărca în memorie.

2. Ceea ce vedeți pe ecran reprezintă rezultatul programului. Puteți produce mai multe rezultate scriind textul pe ecran. Tot ceea ce apare pe ecran de-a lungul execuției programului constituie date de ieşire ale programului.

3. După ce scrieți textul, puteți interacționa cu alte dispozitive. Probabil că veți da o comandă de tipărire a documentului (aproape sigur utilizând opțiunea standard din meniul File, Print) la imprimantă şi îl veți salva pe disc într-un fişier de date.

4. După ce ieşiți din procesorul de text, sistemul de operare preia controlul. Programul de procesare de text nu se mai află în memorie, ci undeva pe disc, în siguranță.

După cum vedeți, rezultatul execuției unui program constituie datele de ieşire. Instrucțiunile sunt cele care produc aceste rezultate, în figura 2 este prezentată o trecere în revistă a procesului program/ieşire. Programele moderne produc datele de ieşire în diverse moduri. Programele redau piese muzicale, comunică cu alte calculatoare prin liniile telefonice şi controlează dispozitivele exterioare. Datele de ieşire trimise pe ecran sau la imprimantă încă mai constituie majoritatea rezultatelor programelor.

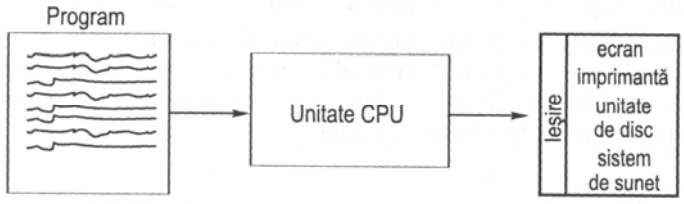


FIGURA 2

Programul vine de pe disc, este executat şi apoi îşi trimite rezultatele spre oricare dintre dispozitivele de ieşire, cum ar fi discul, ecranul sau imprimanta (sau mai multe dintre ele, din acelaşi program).

Atunci când un program se află în memorie, el nu este singur. Sistemul de operare este întotdeauna rezident în memorie. Dacă nu ar fi, nu ați putea nici să încărcați un program de pe disc, nici să-l rulați, deoarece sistemul de operare este cel care încarcă de fapt programele în şi din memorie, atunci când dați comanda corectă. Memoria limitată constituie adeseori o problemă pentru programele mai mari. Trebuie să vă amintiți că un program prelucrează date, iar datele trebuie să se afle în memorie - ca şi programul - înainte ca acesta să le poată prelucra cu uşurință.

În figura 3 se poate vedea cum arată memoria tipică instalată pe calculator atunci când este rulat un program. Sistemul de operare ocupă o porțiune mare, programul trebuie să se afle şi el acolo şi, în final, trebuie să existe loc şi pentru date.

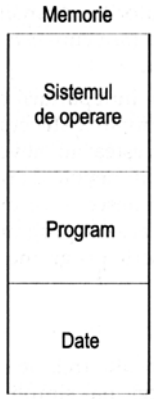


FIGURA 3

Schema tipică a unei memorii arată că sistemul de operare împarte memoria cu programele aflate în execuție.

Cu cât are o memorie mai mare PC-ul dumneavoastră, cu atât mai rapid sunt rulate programele. Memoria suplimentară se traduce prin aceea că sistemul de operare va trebui să comute mai rar pe disc, înainte şi înapoi, pe măsură ce rulează programul. Unele programe sunt chiar complet conținute în memorie, tot timpul cât este rulat programul. Ca programator, va trebui să lucrați cu memorii mai mari, cum ar fi 32 MB sau 64 MB, astfel încât să vă puteți testa în mod adecvat programele rulate, menținând simultan mediul de programare încărcat.

O controversă adeseori întâlnită în literatura referitoare la calculatoare se referă la întrebarea dacă programarea este o artă sau o ştiință. De-a lungul anilor au existat progrese în programare care, dacă sunt utilizate, îmbunătățesc exactitatea programului, posibilitatea acestuia de a fi citit şi de a fi *întreținut* (procesul de modificare ulterioară a programului, pentru a putea îndeplini sarcini diferite sau suplimentare). Multe dintre aceste progrese nu reprezintă decât sugestii, ceea ce înseamnă că programatorii nu sunt obligați să le utilizeze pentru a scrie programe ce funcționează corect.

Două dintre cele mai importante progrese din programare sunt mai mult de natură filosofică decât inginerească. Acestea sunt *programarea structurată* şi *programarea orientată spre obiecte.* Aceste două progrese din programare vor fi analizate pe larg în capitolele ce urmează. Ambele sugerează modalități prin care un programator poate scrie programe mai bune. Pe de altă parte, acestea constituie doar sugestii de abordare a programării; programatorii le pot ignora (şi mulți o fac).

Există multe modalități de a scrie chiar şi cele mai mici şi mai simple programe. Aşa cum scriitorii şi muzicienii scriu şi interpretează diferit, şi programatorii pot avea propriul stil. Deci, s-ar părea că programarea este mai mult o artă decât o ştiință. Pe o scală continuă de la ştiință la artă, cei care sunt de această părere s-ar situa mai aproape de adevăr decât puținii care cred că programarea este mai mult o ştiință. Oricum ar fi, pe măsură ce se fac mai multe progrese în dezvoltarea unor strategii ca programarea structurată şi programarea orientată spre obiecte, se constată o deviere a modului de gândire. Cu proliferarea masivă a calculatoarelor din zilele noastre, instruirea intensă în scopul pregătirii programatorilor *de* mâine este în curs de desfăşurare. Deoarece industria programării este tânără (aşa cum este întreaga industrie a calculatoarelor), mai sunt încă multe progrese de făcut.

Unii dintre cei mai fideli partizani ai îndepărtării de strategia artistică, în direcția unei abordări ştiințifice - utilizând programarea structurată şi orientată spre obiecte - sunt firmele ce plătesc programatorii. Acestea au nevoie de reacții rapide la modificările condițiilor din lumea afacerilor, deci au nevoie de programe scrise cât mai repede şi mai exact posibil. Pe măsură ce se fac noi descoperiri în programare, din ce în ce mai multe firme sunt pe cale să adopte politici care cer programatorilor să utilizeze metode cât mai ştiințifice şi mai bătătorite pentru a scrie programe mai bune.

**4. Cunoaşterea limbajului**

Instrucțiunile pe care le dați calculatorului trebuie să fie exprimate într-un limbaj pe care acesta îl poate înțelege. La nivelul cel mai simplist, un calculator nu reprezintă decât câteva mii *de* întrerupătoare ce se închid şi deschid fulgerător de rapid. Un întrerupător se poate afla doar într-una din două stări: *închis* sau *deschis.* Deoarece oricare dintre aceste două stări electrice poate fi cu uşurință controlată de întrerupătoare electronice, multe mii de asemenea întrerupătoare controlează ceea ce face calculatorul de la o microsecundă la alta.
Dacă ar fi după calculatorul dumneavoastră, ar trebui să-i dați instrucțiuni utilizând întrerupătoare ce reprezintă stările electrice de „închis" sau „deschis" . De fapt, aceasta este exact modalitatea în care se programau primele calculatoare. Pentru introducerea tuturor programelor şi datelor trebuia utilizat un panou de întrerupătoare.

La nivelul de bază al calculatorului, stările electrice de „închis" şi „deschis" sunt reprezentate prin cifrele 1 sau 0. Puteți controla ce face calculatorul dacă cunoaşteți tiparul corect de cifre 1 şi 0 necesar pentru a-i da comenzi. Puteți programa un calculator printr-o serie de l şi O, dacă dispuneți de instrumentele de sistem corecte. Desigur că programarea cu 1 şi 0 nu este cu mult mai bună decât mutarea întrerupătoarelor în sus sau în jos pe panoul de întrerupătoare, aşa că trebuie să existe şi o cale mai bună.

În ciuda celor văzute probabil în filmele SF, în viitorul apropiat calculatoarele nu vor fi capabile să învețe vreo limbă vorbită de către oameni. Trebuie să învățați un limbaj de programare, dacă doriți să faceți calculatorul să îndeplinească sarcinile dorite.

Limba engleză şi toate celelalte limbi vorbite sunt prea ambigue pentru calculatoare. Greierele umane pot descifra propozițiile în mod intuitiv, ceea ce maşinile nu pot face. Există câteva încercări de creare a *inteligenței artificiale* - ştiința de a programa calculatoarele astfel încât acestea să poată fi capabile de a învăța singure. Aici este inclusă şi programarea lor pentru a putea înțelege o limbă vorbită, cum ar fi engleza, în ciuda progreselor recente, inteligența artificială se află încă la mulți ani distanță în viitor (dacă este cumva posibil pentru calculatoare să înțeleagă comenzi simple în engleză).

Prin urmare, calculatoarele şi oamenii se află la capetele opuse ale spectrului. Oamenii doresc să vorbească limba lor, dar aşa fac şi calculatoarele ce nu înțeleg, de fapt, decât cifrele 1 şi 0. Trebuie să existe o cale de mijloc. Limbajele de programare au fost create pentru a încerca să împace atât calculatorul, cât şi persoana ce-l programează. Limbajele de programare utilizează cuvinte similare celor folosite de către oameni, dar acestea au o sintaxă cu totul deosebită (structură, ordine, gramatică şi ortografie), ce lasă foarte puțin loc ambiguităților - atât de răspândite în limba vorbită. Calculatorul poate prelua limbajul de programare şi-l poate traduce în limbaj maşină, compus din 1 şi 0; programatorul poate învăța, reține şi utiliza limbajele de programare mai eficient decât limbajul compus din 1 şi 0, deoarece acestea sunt similare limbii vorbite, cu toate că sunt mai precise şi mai simple.

În decursul acestui curs veți afla multe lucruri despre diverse limbaje de programare. Este posibil să fi auzit deja de unele dintre ele, iar de altele se poate să nu fi auzit. De-a lungul anilor ce au trecut de la inventarea primului limbaj de calculator, au fost scrise multe limbaje de programare, dar numai câteva dintre ele au întâietate față de celelalte. Iată o listă cu câteva limbaje de programare, care au căpătat oarecare notorietate de-a lungul timpului:

Limbaj maşină – Assembler, Algol, PL/1, PROLOG, LISP, COBOL, Forth, RPG, RPG II, Pascal, Object Pascal, C, C++, SNOBOL, ADA, Fortran, Basic, Visual Basic, Small Talk, APL, Java

Fiecare limbaj de programare are propriile dialecte.

Calculatorul este ca un turn Babei modern, responsabil indirect de existența mai multor limbaje de programare decât au fost vreodată necesare. Majoritatea celor considerați a fi „experți în calculatoare" s-ar putea să cunoască numai câteva dintre ele - între două şi cinci - şi probabil că ştiu numai unul sau două foarte bine. Deci, numărul mare de limbaje nu trebuie să vă împiedice de a învăța programare.

Sunt mai multe motive pentru care există atât de multe limbaje de programare. Oamenii au diverse preferințe, unele limbaje sunt mai bune decât altele pentru anumite sarcini, iar unele persoane au acces doar la unul sau două limbaje.

Programul pe care-l scrieți se numeşte *program sursă* (sau *cod sursă).* Atunci când veți întâlni pe parcursul acestei cărți termenul de *program sursă,* veți şti că se referă la program înainte de a fi compilat.

**Translatorul de limbaj**

De fapt, calculatorul nu poate înțelege nici limbajele BASIC, C, Pascal sau oricare dintre limbajele de programare. S-ar putea ca aceasta să vă provoace nedumerire, pentru că în secțiunea precedentă s-a explicat

că limbajele de calculator sunt importante tocmai pentru că aceste calculatoare nu pot înțelege engleza sau orice altă limbă vorbită de către oameni. Chiar aşa fiind, calculatorul nu înțelege nici limbajul BASIC - dar un program poate traduce cu uşurință limbajul BASIC într-o serie de cifre 1 şi 0, pe care calculatorul le *poate* înțelege.

Există două feluri de translatoare de limbaj: *compilatoarele* şi *interpretoarele.* Ambele iau limbajul de programare respectiv, cum ar fi Pascal, şi-l traduc într-o formă ce poate fi citită de către calculator. Fiecare utilizează o strategie diferită, dar rezultatele sunt aceleaşi; ambele iau programele şi le transformă într-o serie de cifre 1 şi 0 (denumită *limbaj maşină),* aşa cum se arată în figura 4.

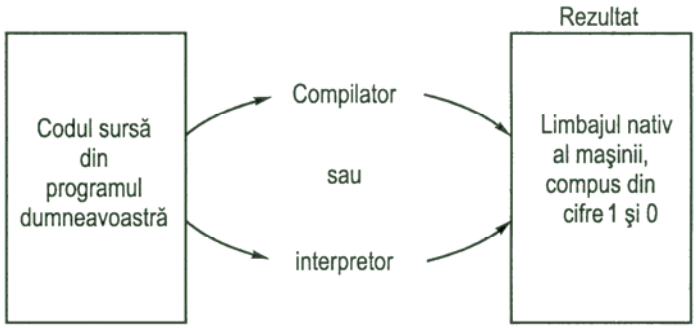


FIGURA 4

*Atât interpretoarele, cât şi compilatoarele traduc codul sursă în limbajul maşină, de nivel inferior, pe care-l poate înțelege calculatorul.*

**Interpretoarele**

Unele limbaje de programare, cum ar fi APL şi BASIC, există ca interpretoare (există şi compilatoare asemănătoare cu interpretorul BASIC, cum ar fi Visual Basic). Interpretoarele traduc câte o linie o dată, executând fiecare linie pe măsură ce este tradusă. Denumirea de *interpretor* este foarte descriptivă. Se poate cunoaşte modul în care operează interpretoarele calculatoarelor deoarece se cunoaşte modul în care lucrează interpreții umani.

Să presupunem că vă dă cineva o carte într-o limbă străină pe care n-o cunoaşteți. Pentru a înțelege conținutul cărții, ați putea angaja un interpret uman care să v-o citească. Interpretul citeşte un rând, îl traduce, apoi vă citeşte următorul rând. Singurul dezavantaj al interpretoarelor (spre deosebire de compilatoare, descrise în secțiunea următoare) este că interpretorul trebuie să traducă din nou liniile pe care doriți să le citească încă o dată. Interpretarea este mai lentă decât compilarea, în figura 5 este ilustrată poziția interpretului uman între dumneavoastră şi carte. Rețineți aceasta atunci când citiți despre interpretoarele calculatoarelor.

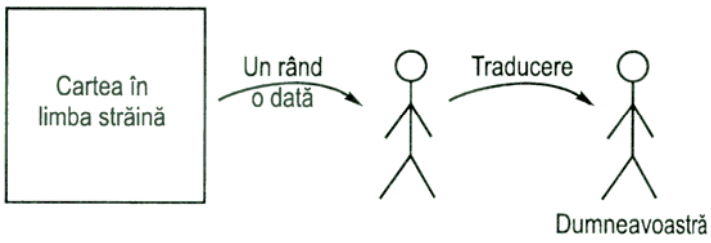


FIGURA 5

*Un interpretor traduce pe rând câte o linie.*

Un interpretor de limbaj de programare acționează ca şi un interpret uman, prin aceea că procesul de interpretare a câte unei linii o dată poate fi lent. Adeseori, calculatoarele îndeplinesc sarcini repetitive, cum ar fi

tipărirea unor sute de cecuri de pe statul de plată. Dacă aveți un limbaj de programare cu interpretor, acesta ia fiecare linie a programului, o traduce, apoi o execută. Acest proces face ca sarcinile repetitive să decurgă foarte încet.

Mai jos este prezentată o porțiune dintr-un program în BASIC. Tot ceea ce trebuie să ştiți este că aceste linii se repetă de 50 de ori:

```
FOR i =  1   TO 50
PRINT  i
NEXT  i
```

Motivul pentru care interpretorul este lent constă în faptul că - de fiecare dată când se repetă o linie - trebuie s-o interpreteze din nou. Rezultă că aceste trei linii de program sunt interpretate şi executate de 50 de ori.

Interpretoarele prezintă şi avantaje față de compilatoare. Limbajele interpretate sunt puțin mai uşor de învățat de către programatorii începători. Din punct de vedere istoric, procesul de compilare a programelor este mai dificil decât interpretarea lor (cu toate că multe dintre compilatoarele actuale sunt aproape la fel de uşor de utilizat ca şi interpretoarele). Timpul dintre rularea unui program până la obținerea rezultatelor acestuia este mai scurt, atunci când se utilizează un interpretor. Această reacție rapidă este utilă începătorilor în programare.

**Compilatoarele**

În loc de a angaja un interpret pentru a vă traduce cartea scrisă într-o limbă străină, ați putea mai bine să-i cereți traducătorului să *compileze* cartea pentru dumneavoastră. Traducătorul poate să stea jos şi să scrie interpretarea pentru dumneavoastră, fără să v-o citească în timpul acesta. Cu toate că această metodă necesită un timp mai îndelungat, dumneavoastră puteți citi versiunea compilată şi puteți face referiri la aceasta de câte ori este necesar.

În figura 6 este prezentat procesul de compilare a unei cărți. Partea bună în a avea cartea compilată este că, o dată aceasta compilată, nu mai este nevoie de interpret.

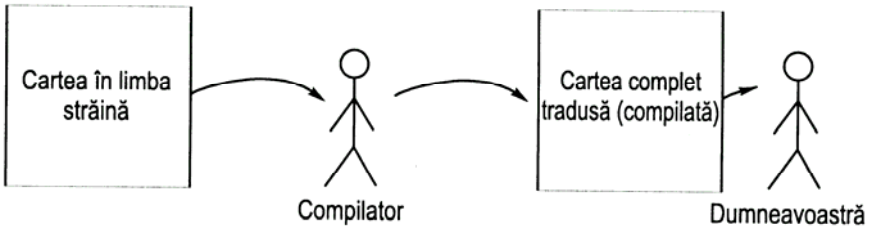


FIGURA 6

După compilarea cărții, nu mai este nevoie de traducător.

Compilarea unui program necesită un pas în plus față de interpretarea acestuia. Trebuie să-i cereți compilatorului să compileze programul şi apoi să aşteptați rezultatul. Numai după compilarea întregului program va putea funcționa vreo parte din acesta. Majoritatea programatorilor începători întâmpină destule dificultăți în învățarea interacțiunilor programului cu componentele hardware, sistemului de operare şi limbajului în sine, fără a mai avea de învățat şi comenzile compilatorului.

Majoritatea firmelor preferă limbaje de programare cu compilatoare deoarece - o dată compilate - programele sunt executate mult mai rapid decât dacă ar fi interpretate. In exemplul din limbajul BASIC de mai sus, cele trei linii sunt compilate în limbaj maşină o singură dată. După ce este terminată compilarea, programul execută cele 50 de repetiții fără traducere.

Firmele preferă programele compilate față de cele interpretate şi din motive de securitate. O dată compilat, un program nu poate fi modificat cu uşurință. Atunci când cumpărați un program, sunt şanse bune ca acesta să fie deja compilat. Puteți să-l rulați, dar nu puteți vedea codul sursă.

Unele dintre limbajele de programare actuale, cum ar fi Visual Basic, permit atât rularea programelor interpretate, cât şi a celor compilate. Prin urmare, se obțin atât avantajele oferite de către un limbaj interpretat privind testarea, cât şi - după terminarea testării - viteza şi securitatea oferite de către limbajele cu compilatoare.

**Exactitatea este totul**

Acum vă dați perfect de bine seama că un calculator este o maşină ce nu poate trata cu succes ambiguitățile. Pacostea unui programator o constituie seria de erori ce apar în program. Programatorii trebuie să se asigure că nu scriu programe ce conțin erori, cu toate că acest lucru nu este întotdeauna aşa de uşor cum ar putea părea. În terminologia folosită în domeniul calculatoarelor, o eroare de program este cunoscută sub denumirea de *bug.* Atunci când programatorii fragmentează problema în instrucțiuni detaliate, adeseori se întâmplă ca aceştia să omită unele lucruri sau să programeze ceva ce nu trebuie. Atunci când este rulat programul, se strecoară erori din cauza bug-urilor din acesta.

***Depanarea*** este procesul pe care-l parcurge programatorul pentru a elimina erorile dintr-un program. Atunci când acesta scrie un program, adeseori îl rulează în forma sa neterminată (atât cât se poate), pentru a detecta cât mai multe erori posibil şi a le exclude din programul final. Adeseori, majoritatea erorilor pot fi găsite numai după ce programul este scris în întregime.

De multe ori, programatorii începători nu înțeleg cât de uşor este să introduci hibe în program. Aşteptați-vă la ele şi nu vă lăsați surprins. Mulți studenți în domeniul programării duc programul instructorului, spunând: „calculatorul nu merge cum trebuie", când - în realitate - programul conține una sau două hibe. Atunci când începeți să scrieți propriile programe, aşteptați-vă să aveți de corectat unele probleme. Nimeni nu poate scrie întotdeauna un program perfect.

În funcție de lungimea programului, timpul necesar programatorului (sau programatorilor) pentru a corecta problemele din acesta este adeseori aproape la fel de lung ca şi cel necesar pentru scrierea programului. Unele erori sunt foarte greu de găsit.

Există două categorii de hibe în programe: erori de sintaxă şi erori de logică. Pentru a vedea care este diferența, încercați să găsiți cele două erori din următoarea afirmație:

*Există două errori în această propoziție.*

Aveți nevoie de un indiciu? încă nu; mai încercați să găsiți cele două erori, înainte de a merge mai departe. Prima eroare este evidentă. Cuvântul *„errori"* este scris greşit; ar trebui să fie *„erori".* A doua problemă este mult mai dificil de găsit. A doua greşeală din această afirmație constă în faptul că întreaga premisă pe care se bazează afirmația este incorectă. Există o *singură* eroare în această afirmație, iar aceasta constă în scrierea greşită a cuvântului *„errori".* Deci, logica întregii afirmații constituie ea însăşi o eroare.

Acest exemplu demonstrează diferența dintre o *eroare de sintaxă* şi o *eroare de logică.* Eroarea de sintaxă este mult mai uşor de identificat. De obicei, erorile de sintaxă constau în comenzi din limbajul de programare ortografiate greşit şi probleme de gramatică legate de modul de utilizare a limbajului de programare. Erorile de logică apar atunci când programul dumneavoastră este corect din punct de vedere al sintaxei, dar i-ați spus să facă ceva ce nu reprezintă ceea ce ar trebui făcut de fapt.

Compilatoarele şi interpretoarele pot localiza erorile de sintaxă din programul dumneavoastră atunci când încercați să-l compilați sau să-l rulați. Acesta este un alt motiv pentru care erorile de sintaxă sunt mai uşor de depistat: calculatorul vă spune unde se află. Atunci când un calculator întâlneşte o eroare de sintaxă, se opreşte şi refuză să analizeze mai departe programul până când aceasta nu este corectată.

Să presupunem că scrieți un program de tipărire a facturilor pentru clienții firmei dumneavoastră. Din *cauza* unei erori, calculatorul tipăreşte toate facturile cu un deficit de 1.000$. Cu alte cuvinte, conform facturilor fiecare client are un credit de 1.000$. Calculatorul şi-a făcut treaba, acționând conform instrucțiunilor din programul dumneavoastră. Este evident că acesta nu a conținut erori de sintaxă, deoarece a fost rulat fără să se oprească. Totuşi, erorile de logică îl împiedică să calculeze corect.

Testarea extensivă este de importanță majoră. Programatorul doreşte să elimine toate erorile, astfel încât programul să funcționeze corect în final, atunci când va fi folosit de către utilizator. Cu cât este mai vast programul, cu atât această sarcină este mai dificilă, eliminarea bug-urilor din programe constituie doar o parte din sarcinile la care se înhamă zilnic programatorul.

Mai există încă un tip de eroare: *eroarea din timpul execuției.* De fapt, erorile din timpul execuției sunt aproape întotdeauna cauzate de către o greşeală de logică, datorată faptului că programatorul nu a reuşit să prevadă - deci să rezolve - o problemă potențială. Erorile din timpul execuției pot apărea dacă un program încearcă să scrie pe o dischetă fără a verifica mai întâi dacă clapeta unității de dischetă este închisă şi dacă discheta se află în interiorul unității. O eroare din timpul execuției poate apărea, de exemplu, dacă programul face o împărțire la zero (împărțirea la zero nu are sens din punct de vedere matematic). Cu cât veți programa mai mult, cu atât veți învăța mai bine să prevedeți potențialele erori din timpul execuției.

**5.Proiectarea unui program**

Programatorii învață să aibă răbdare încă de la începutul carierelor lor în programare. Ei descoperă că proiectarea corectă este decisivă pentru un program de succes. Poate că ați auzit termenul de *analiză şi proiectare de sistem.* Aceasta este denumirea atribuită practicii de analizare a problemei şi de proiectare ulterioară a programului, conform acestei analize. Au fost scrise cursuri de facultate şi cărți întregi despre analiza şi proiectarea de sistem. Această lecție va aborda ideile principale, permițându-vă să vedeți prin ce trece un programator de calculator mainstream înainte de a scrie programe.

În această lecție veți afla răspunsurile la următoarele chestiuni:

• De ce este atât de importantă proiectarea programelor;
• Care sunt cele trei etape necesare în scrierea unui program;
• Ce se înțelege prin definirea datelor de ieşire;
• De ce este mai bună proiectarea de sus în jos decât cea de jos în sus;
• Ce este o organigramă;
• Ce semnificație au simbolurile dintr-o organigramă;
• Când este mai avantajoasă utilizarea pseudo-codului decât a organigramei;
• Care este etapa finală în procesul de programare.

**Necesitatea proiectării**

Atunci când un constructor începe să construiască o casă, nu ia un ciocan şi se apucă să înalțe pereții bucătăriei. Înainte de a începe ceva, arhitectul trebuie să proiecteze casa cea nouă. Aşa cum veți vedea în curând, şi un program trebuie proiectat înainte de a fi scris.

Un constructor trebuie să ştie mai întâi ce doreşte cumpărătorul casei. Nimic nu poate fi construit decât atunci când constructorul are în minte rezultatul final. Deci, cumpărătorii casei trebuie să consulte un arhitect. Ei

îi spun acestuia cum vor să arate casa. Arhitectul îi ajută să se hotărască, spunându-le ce este posibil şi ce nu. În stadiile inițiale, prețul constituie întotdeauna un factor care necesită acorduri de compromis între proiectanți şi cumpărători.

După ce arhitectul a conceput planurile casei, constructorii trebuie să planifice resursele necesare pentru construirea acesteia. Numai după terminarea proiectului casei, întocmirea dosarului cu autorizații, plasarea banilor, cumpărarea materialelor şi angajarea lucrătorilor poate începe construcția fizică a casei. De fapt, cu cât constructorul investeşte mai multe eforturi în aceste etape preliminare, cu atât mai repede poate fi terminată construcția.

Dacă se construieşte casa înainte de a fi proiectată adecvat, problema este că eventualii cumpărători ar putea dori să facă modificări atunci când este prea târziu pentru a mai fi posibil. Este foarte dificil de adăugat o cameră de baie între două dormitoare *după* terminarea casei. Ideea principală este de a face proprietarii şi constructorii să se pună de acord asupra aspectului final al casei. Atunci când toate părțile implicate au căzut de acord asupra tuturor specificațiilor, mai rămâne puțin loc pentru ulterioare dezacorduri. Cu cât sunt mai clare planurile inițiale, cu atât se pot ivi mai puține probleme pe parcurs, deoarece toate părțile s-au pus de acord asupra aceloraşi planuri ale casei.

**Proiectarea programelor**

Dumneavoastră trebuie să aveți totdeauna în minte asemănările cu construcția unei case, înainte de a scrie un program de lungime mai mare. Nu trebuie să vă repeziți la tastatură şi să vă apucați să scrieți instrucțiuni din program înainte de a-l proiecta, aşa cum un constructor nu trebuie să ia ciocanul în mână înainte de a fi stabilite planurile casei. Cu cât veți depune mai multe eforturi pentru proiectarea inițială, cu atât veți termina mai repede programul final.

Mulțumită tehnologiei calculatoarelor, un program de calculator este mai uşor de modificat decât o casă. Dacă ați omis o rutină dorită de utilizator, puteți s-o adăugați ulterior mult mai uşor decât ar putea un constructor să adauge o cameră unei case terminate. Cu toate acestea, a adăuga ceva într-un program nu este niciodată atât de simplu ca a-l proiecta corect de la început.

Întreținerea programului, ce are loc după ce acesta este scris, testat şi distribuit, constituie unul dintre aspectele ce consumă cel mai mult timp în procesul de programare. Programele sunt în permanență modernizate, pentru a reflecta noile cerințe ale utilizatorului. Uneori, dacă programul nu este proiectat corect înainte de a fi scris, utilizatorul nu-l va accepta până ce acesta nu va efectua exact ceea ce doreşte el.

Consultanții din domeniul calculatoarelor învață repede să obțină acordul utilizatorului şi chiar semnătura acestuia pe proiectul programului, înainte de a începe programarea. Dacă atât utilizatorul, cât şi programatorul se pun de acord asupra a ceea ce este de făcut, la prezentarea programului mai rămâne puțin loc pentru dispute. Şi firmele cu departamente de prelucrare a datelor cer echipelor de programatori să ajungă la un acord scris cu utilizatorii. Resursele firmei sunt limitate; nu este timp pentru a adăuga ulterior ceva ce trebuia să fie acolo încă de la început. O mare parte din această lecție este dedicată explicării modului în care este bine să fie începută realizarea proiectelor asupra cărora dumneavoastră şi utilizatorul veți cădea de acord.

Există trei etape care trebuie parcurse în scrierea un program:

1. Stabilirea definirii datelor de ieşire;
2. Dezvoltarea raționamentului necesar pentru obținerea acelor date de ieşire;
3. Scrierea programului.

Remarcați că scrierea programului este *ultima* etapă în realizarea acestuia. Nu e atât de absurd cum pare. Amintiți-vă că realizarea fizică a casei este ultima etapă a construcției acesteia; planificarea corectă, înainte de începerea construcției propriu-zise, este de importanță majoră. De fapt, veți descoperi că scrierea

liniilor de program constituie una dintre cele mai uşoare etape ale procesului de programare. Dacă proiectul dumneavoastră este bine gândit, programul se scrie practic singur; scrierea acestuia devine aproape o urmare firească a întregului proces. În restul acestei lecții vor fi analizate aceste trei etape ale proiectării programelor.

**Etapa I: definirea datelor de ieşire**

Înainte de a începe un program, trebuie să aveți clar în minte ce trebuie să facă acesta. Dacă priviți încă o dată modelul fundamental al programării din figura 7 veți observa că ieşirea este ultimul lucru realizat, dar *primul lucru ce trebuie proiectat.* Aşa cum constructorul trebuie să ştie cum va arăta casa înainte de a începe construcția acesteia şi programatorul trebuie să ştie care vor fi datele de ieşire, înainte de a începe scrierea programului.


FIGURA 7

La nivel fundamental, prelucrarea datelor transformă datele în informații.

Ieşirea într-un program înseamnă mai mult decât tipărirea informațiilor. Orice este produs de către program şi este văzut de către utilizator este considerat dată de ieşire iar dumneavoastră trebuie s-o proiectați. Trebuie să ştiți cum trebuie să arate fiecare ecran din program şi ce va fi pe fiecare pagină a fiecărui raport tipărit de acesta.

Unele programe produc o cantitate imensă de date de ieşire. Nu săriți peste această primă etapă atât de importantă a procesului de proiectare, numai pentru că sunt multe date de ieşire. Fiind un rezultat mai complex, devine mai important ca dumneavoastră să-l definiți. Stabilirea definirii datelor de ieşire este relativ facilă - uneori chiar de-a dreptul plictisitoare şi consumatoare de timp. Timpul necesar pentru definirea datelor de ieşire poate fi tot atât de lung ca acela necesar pentru a treia etapă, care constă în scrierea programului (totuşi, cea de-a doua etapă necesită *cel mai mult* timp). Veți consuma tot atât de mult timp - şi chiar mai mult - dacă evitați stabilirea definiri datelor de ieşire în faza de început.

Unul dintre avantajele oferite de sistemul de operare Windows constă în natura sa vizuală. Înainte de apariția acestuia, instrumentele de programare se limitau la proiectarea şi implementarea bazate pe text. Astăzi, proiectarea ecranului unui utilizator înseamnă lansarea unui mediu de programare, cum ar fi Visual Basic, desenarea ecranului şi tragerea obiectelor cu care va interacționa utilizatorul în interiorul acestuia, cum ar fi un buton OK. Prin urmare, puteți proiecta rapid *ecrane prototip,* pe care să le trimiteți utilizatorului. După ce acesta va vedea ecranul cu care va trebui să interacționeze, îşi va putea da seama mult mai bine dacă dumneavoastră ați înțeles care sunt cerințele pe care trebuie să le îndeplinească programul.

Definirea datelor de ieşire înseamnă mai mult decât o proiectare preliminară. Ea permite o bună cunoaştere a acelor elemente ale datelor pe care trebuie să le urmărească, să le calculeze şi să le producă programul. Definirea datelor de ieşire ajută şi la stabilirea tuturor datelor de intrare necesare pentru producerea datelor de ieşire.

Definirea datelor de ieşire constă în multe pagini de detalii. Trebuie să fiți capabil de a specifica toate detaliile unei probleme, înainte de a şti ce rezultat este necesar. Una dintre cele mai bune strategii în specificarea detaliilor unei probleme este proiectarea de sus în jos.

**Proiectarea programelor de sus în jos**

Cea mai bună modalitate de proiectare existentă este cea de sus în jos. Cu ajutorul proiectării de sus în jos, se pot obține detaliile necesare pentru îndeplinirea unei sarcini de programare. *Proiectarea de sus în jos* este procesul de fragmentare a problemei generale în mai multe detalii, până la epuizarea tuturor acestora.

Problema în ceea ce priveşte proiectarea de sus în jos este că programatorii au tendința de a nu o utiliza. Ei au tendința de a proiecta începând din direcția opusă (denumită *proiectare de jos în sus).* Prin ignorarea proiectării de sus în jos, vă încărcați cu sarcina grea de a vă aminti fiecare detaliu care va fi necesar; prin proiectarea de sus în jos, detaliile se desprind de la sine. Dacă utilizați strict proiectarea de sus în jos, nu trebuie să vă preocupe cele mai mici detalii deoarece acest proces se ocupă de producerea acestora.

Unul dintre elementele cheie din proiectarea de sus în jos constă în aceea că vă forțează să amânați detaliile pe mai târziu. Proiectarea de sus în jos vă forțează să gândiți cât de mult posibil în termeni privind problema generală. Proiectarea de sus în jos vă menține concentrat. Prin utilizarea proiectării de jos în sus, este foarte uşor să pierdeți din vedere pădurea din cauza copacilor. Ajungeți prea repede la detalii şi pierdeți din vedere obiectivele generale ale programului.

**Proiectarea de sus în jos în realitate**

Proiectarea de sus în jos poate fi înțeleasă mai uşor prin aplicarea sa la o problemă uzuală din realitate, înainte de a aborda o problemă de calculatoare. Proiectarea de sus în jos nu se aplică doar problemelor de programare. O dată stăpânită, proiectarea de sus în jos se poate aplica în orice situație concretă ce necesită o planificare în detaliu. Probabil cel mai detaliat eveniment pe care-l poate planifica o persoană este o căsătorie. Deci, o căsătorie reprezintă exemplul perfect de observare a planificării de sus în jos în acțiune.

Care este primul lucru ce trebuie făcut pentru a avea loc o căsătorie? Mai întâi, trebuie găsit un viitor partener (pentru a găsi ajutor în această privință vă trebuie o altă carte). Când se ajunge la momentul în care este necesară planificarea căsătoriei, proiectarea de sus în jos constituie cea mai bună strategie de abordare a evenimentului. Modalitatea în care *nu* trebuie planificată o căsătorie este să începi prin a-ți bate capul cu detaliile; totuşi, aceasta este utilizată de către majoritatea oamenilor, încep să se gândească la rochii, la cel care va cânta la orgă, la flori şi la tortul ce va fi servit la recepție. Cea mai mare problemă, atunci când se încearcă cuprinderea tuturor acestor detalii încă de la început, constă în aceea că se pierd din vedere atât de multe; este foarte uşor să uiți un detaliu până când este prea târziu. Detaliile proiectării de jos în sus constituie un obstacol.

**Etapele proiectării de sus în jos**

Cele trei etape necesare în proiectarea de sus în jos sunt:

1. Stabilirea scopului general.
2. Fragmentarea scopului în două, trei sau mai multe detalii. Prea multe detalii duc la omisiuni.
3. Evitarea detaliilor inutile pe cât posibil. Trebuie repetate etapele l şi 2, până când fragmentarea problemei nu mai are sens.

Care este scopul general al unei căsătorii? Gândind în cei mai generali termeni posibili, „a face nunta" este o expresie cât se poate de generală. Dacă ați fi însărcinat cu planificarea unei căsătorii, scopul general de „a face nunta" v-ar îndrepta direct spre țintă. Să presupunem că, la cel mai înalt nivel, scopul este „a face nunta".

Scopul general vă menține concentrat, în ciuda naturii sale prolixe, „a face nunta" omite asemenea detalii cum ar fi planificarea lunii de miere. Dacă nu stabiliți exact limitele problemei la care lucrați, vă veți pierde în detalii şi - ceea ce este mai important - veți uita o parte dintre acestea. Dacă planificați atât nunta, cât şi luna de

miere, trebuie să faceți două proiecte de sus în jos sau trebuie să includeți călătoria din luna de miere în scopul general, aflat la cel mai înalt nivel. Acest plan al nunții include căsătoria în sine - ceremonia şi recepția - fără a conține şi detalii legate de luna de miere.

Acum, că ştiți încotro vă îndreptați, începeți să fragmentați scopul general în două sau trei detalii. Să zicem: culorile care se vor purta la nuntă, lista de invitați, plata preotului, ... *hopa,* sunt prea multe detalii! Ideea de bază în proiectarea de sus în jos este să laşi deoparte detaliile, cât de mult este posibil. Nu vă grăbiți. Atunci când descoperiți că ați fragmentat problema curentă în mai mult decât trei sau patru părți, înseamnă că v-ați grăbit în proiectarea de sus în jos. Lăsați deoparte detaliile, în esență, puteți fragmenta scopul de „a face nunta" în următoarele două componente principale: ceremonia şi recepția.

**Utilizarea proiectării de sus în jos**

Următoarea etapă în proiectarea de sus în jos constă în a face acelaşi lucru cu fiecare dintre aceste componente. Ceremonia se compune din participanți şi amplasare. Participanții la ceremonie includ oaspeții, petrecerea şi lucrătorii (preotul, organistul şi aşa mai departe - dar aceste detalii vor veni puțin mai târziu).
Desigur că proiectarea de sus în jos produce un rezultat de formă triunghiulară, ale cărui părți superioare sunt prezentate în figura 8. Mai sunt încă multe detalii de adăugat, dar aceasta este ideea generală; trebuie să lăsați deoparte detaliile cât de mult este posibil. Detaliile apar de la sine, pe măsură ce fragmentați sarcinile în mai multe componente. Asigurându-vă că scopul suprem conține ideea generală spre care țintiți, detaliile vor veni de la sine.

În curând, nu veți mai avea loc pe pagină. Nu-i nimic - utilizați mai multe coli de hârtie, în proiectarea de sus în jos, puteți să numerotați foile şi să plasați numărul paginii în caseta de „deasupra" acestora. De fapt, evidența foilor nu este atât de importantă cum ați putea crede. Prima pagină este punctul central care vă asigură că vă îndreptați spre scopul dorit. Continuați să fragmentați fiecare detaliu ulterior, până când nu veți mai putea continua, în final, veți descoperi că nu a fost omis nici un detaliu.

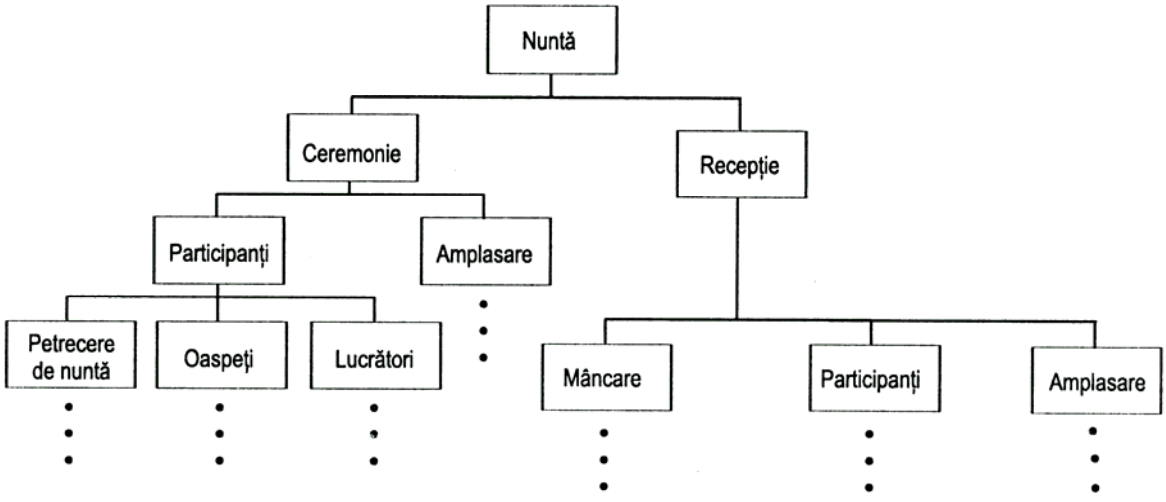


FIGURA 8

Prima parte a planificării unei nunți prin utilizarea proiectării de sus în jos include cele mai importante scopuri.

Deocamdată, nu trebuie să vă preocupe ordinea detaliilor. Scopul proiectării de sus în jos constă în producerea tuturor detaliilor necesare (în final), nu o ordonare oarecare a acestor detalii. Trebuie să ştiți spre ce țintiți şi exact ceea ce este necesar, înainte de a lua în considerare relațiile dintre aceste detalii şi ordinea lor.

În final, veți avea mai multe pagini cu detalii ce nu mai pot fi fragmentate, în acest caz, probabil că veți sfârşi cu mâncarea servită la recepție, cum ar fi arahidele servite la gustări. (Totuşi, dacă începeți să faceți o listă cu aceste detalii, s-ar putea să uitați multe dintre ele.) Trecând la o problemă mai „computerizată", să presupunem că v-ați asumat sarcina de a scrie un program de efectuare a statului de plată pentru o firmă. Ce ar necesita programul de efectuare a statului de plată? Ați putea începe prin a întocmi o listă a detaliilor din statul de plată, cum ar fi aceasta:

• Tipărirea cecurilor din statul de plată;
• Calculul impozitelor;

Ce este greşit în această strategie? Dacă ați spus că detaliile sunt puse prea devreme, aveți dreptate. Locul cel mai bun de pornire este de la vârf. Scopul cel mai general al unui program de realizare a statului de plată ar putea fi „efectuarea statului de plată". Acest scop suprem menține în afara programului alte detalii (nu va fi inclusă prelucrarea nici unui registru general, în afară de cazul în care o parte a sistemului de state de plată modernizează un fişier de registru general) şi vă menține atenția concentrată pe problema din fața dumneavoastră. Priviți figura 9. Aceasta ar putea fi prima pagină din proiectarea de sus în jos a statului de plată. Orice program de stat de plată trebuie să includă un mecanism oarecare de introducere, ştergere şi modificare a informațiilor despre salariați, cum ar fi adresa, oraşul, statul, codul poştal, scutirile de impozite şi aşa mai departe. Ce alte detalii despre fiecare salariat mai sunt necesare? Deocamdată nu întrebați. Proiectul nu este încă gata pregătit pentru toate aceste detalii.

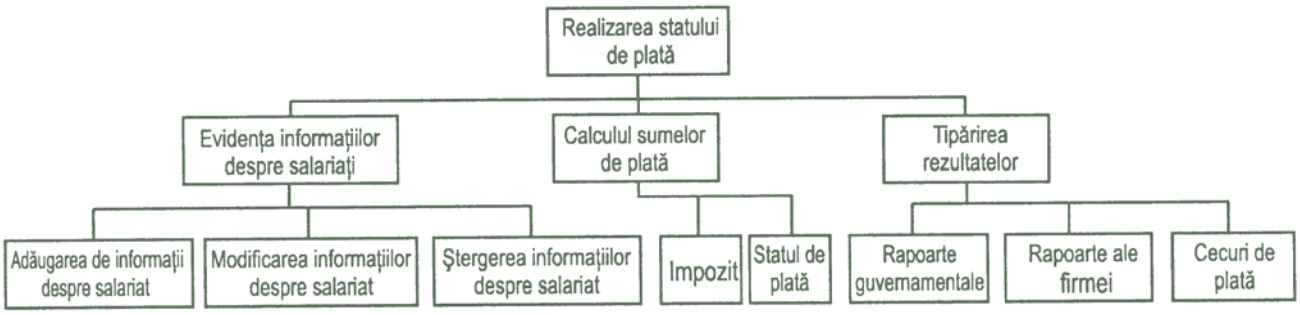


FIGURA 9

Prima pagină reprezentând proiectarea de sus în jos a programului de efectuare a statului de plată ar include detaliile de cel mai înalt nivel.

Mai este mult până ce veți termina proiectarea de sus în jos a statului de plată, dar figura 9 reprezintă primul pas. Trebuie să continuați fragmentarea fiecărei componente, până la apariția detaliilor finale. Numai după ce ați pregătit toate detaliile, puteți începe să stabiliți ce rezultate va produce programul.

Numai după ce ați reuşit, împreună cu utilizatorul, identificarea tuturor detaliilor prin proiectarea de sus în jos, puteți stabili ce vor cuprinde aceste detalii.