Paradigmas de Linguagens de Programação

Tipos de Dados Abstratos

Cristiano Lehrer, M.Sc.

ybadoo

Conceito de Abstração

- O conceito de abstração é fundamental em programação.
- Quase todas as linguagens suportam abstração de processos, através de subprogramas:
  - Exemplo em C:
    - `int soma(int a, int b)`
- Quase todas as linguagens de programação projetadas desde 1980 tem suporte à abstração de dados com algum tipo de módulo.

Encapsulamento

- Motivação original:
  - Grandes programas tem duas necessidades especiais:
    - Algum tipo de organização, além da simples divisão em subprogramas.
    - Algum tipo de compilação parcial:
      - Unidades de compilação são menores que o programa inteiro.
- Solução óbvia:
  - Um agrupamento de subprogramas que são logicamente relacionados em uma unidade que pode ser compilada separadamente:
    - São chamados encapsulamento.

Exemplo de Mecanismos de Encapsulamento

- Subprogramas aninhados em alguma linguagem similar ao ALGOL:
  - Pascal
- FORTRAN 77 e C:
  - Arquivos contendo um ou mais subprogramas podem ser compilados independentemente.
- FORTRAN 90, C++, Ada (e outras linguagens contemporâneas):
  - Separadamente em módulos compiláveis.

Abstração de Dados (1/3)

- Um tipo abstrato de dados (TAD) é um tipo de dados definido pelo usuário que satisfaz a duas condições:
  - A representação e as operações do tipo são definidos em uma unidade sintática simples:
    - Outras unidades podem criar variáveis daquele tipo.
  - A representação do tipo é oculto das unidades do programa que usam esses tipos, assim as únicas operações possíveis são aquelas que provêm da definição do tipo.
    - O código do cliente não depende da implementação do tipo.

Abstração de Dados (2/3)

- Vantagens:
  - Organização do programa, alterabilidade (tudo que estiver associado com uma estrutura de dados está junto), e compilação separado.
  - Confiabilidade:
    - Ao esconder as representações dos tipos, o código do usuário não pode acessar diretamente a implementação do tipo.
    - O código do usuário não pode depender da representação dos dados, permitindo que tal representação seja alterada sem afetar o código do usuário.

Abstração de Dados (3/3)

- Tipos Primitivos (*built-in*) são tipos abstratos de dados:
  - Exemplo: tipo **int** no C
    - A representação é oculta.
    - Todas as operações são *built-in*.
    - Programas do usuário podem definir objetos do tipo **int**.
    - Tipos de dados definidos pelo usuário devem ter as mesmas características dos TAD *built-in*.

Exemplo (1/4)

```
// Arquivo Ponto.h
typedef struct ponto Ponto;

Ponto* create (float, float);

void destroy (Ponto*);

float getX (Ponto*);

float getY (Ponto*);

float distance (Ponto*, Ponto*);
```

Exemplo (2/4)

```
// Arquivo Ponto.c
#include <stdlib.h>
#include <math.h>
#include "Ponto.h"

struct ponto{
    float x;
    float y;
};

Ponto* create (float x, float y){
    Ponto* p = (Ponto*) malloc(sizeof(Ponto));
    if (p != NULL){
        p->x = x;
        p->y = y;
    }
    return p;
}
```


ybadoo

Exemplo (3/4)

```
void destroy (Ponto* p){
    free(p);
}

float getX (Ponto* p){
    return p->x;
}

float getY (Ponto* p){
    return p->y;
}

float distance (Ponto* p1, Ponto* p2){
    float dx = p1->x - p2->x;
    float dy = p1->y - p2->y;

    return sqrt(dx*dx + dy*dy);
}
```


ybadoo

Exemplo (4/4)

```
// Arquivo main.c
#include <stdio.h>
#include "Ponto.h"

int main() {
  Ponto* p = create(10, 21);
  Ponto* q = create(7, 25);

  printf("%f", distance(p, q));

  destroy(p);
  destroy(q);

  return 0;
}
```


ybadoo