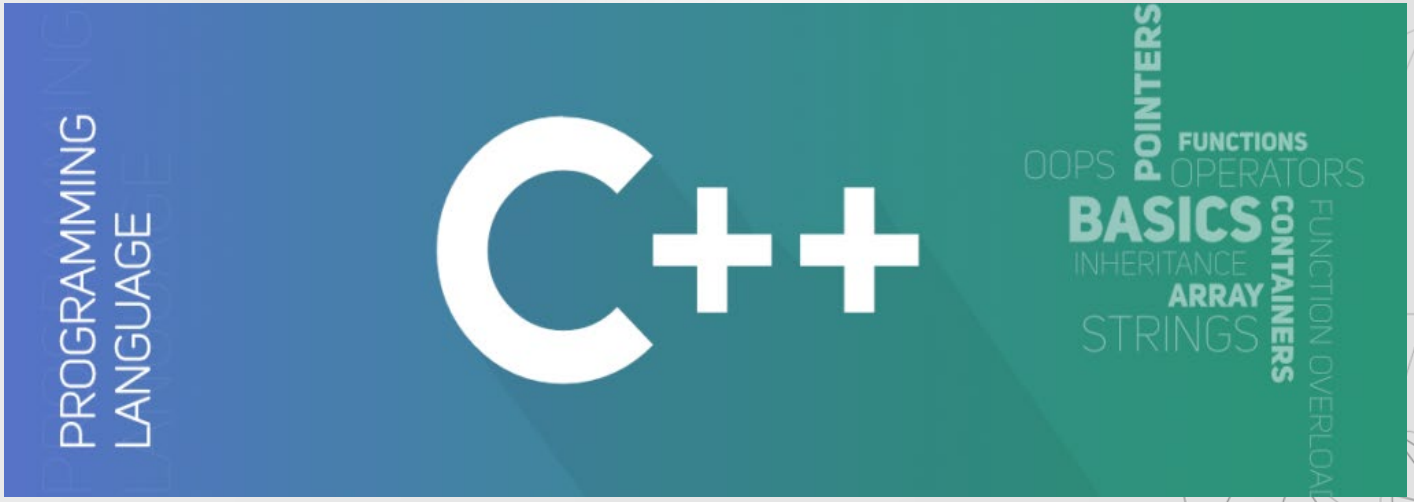
Danh Sách Liên Kết

C++

Vòng lặp là gì?

1. Danh sách liên kết là một cấu trúc dữ liệu được sử dụng để lưu trữ các phần tử tương tự như mảng nhưng có nhiều điểm khác biệt.
2. DSLK có thể mở rộng và thu hẹp một cách linh hoạt.
3. Không lãng phí bộ nhớ nhưng cần thêm bộ nhớ để lưu phần con trỏ.
4. Đây là CTDL cấp phát động nên khi còn bộ nhớ thì sẽ còn thêm được phần tử vào DSLK.



1

So sánh giữa mảng và DSLK:

So sánh giữa mảng và DSLK

(Mảng)

-Ưu điểm:

- Đơn giản và dễ sử dụng.
- Truy cập mảng với độ phức tạp là hằng số.

-Nhược điểm:

- Có thể gây lãng phí bộ nhớ nếu không sử dụng hết bộ nhớ xin cấp phát cho mảng.
- Kích thước mảng là cố định.
- Bộ nhớ cấp phát theo khối.
- Việc chèn và xóa phần tử khó khăn.

(DSLK)

-Ưu điểm:

- Có thể mở rộng với độ phức tạp là hằng số.
- Dễ dàng mở rộng và thu hẹp kích thước.

-Nhược điểm

- Khó khăn trong việc truy cập một phần tử ở vị trí bất kỳ (O(n)).
- Khó khăn trong việc cài đặt.
- Tốn thêm bộ nhớ cho phần tham chiếu bổ sung. Có thể cấp phát số lượng lớn các node tùy vào bộ nhớ.

2

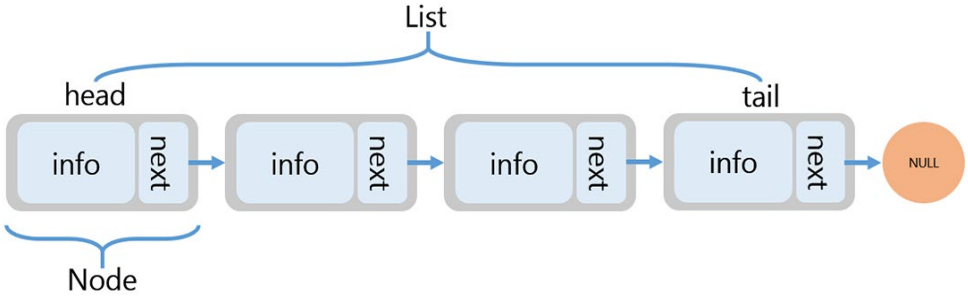
DSLK đơn (Single Linked List)

DSLK đơn:

DSLK bao gồm một số các node trong đó mỗi node có con trỏ next tới node tiếp theo nó trong DSLK. Liên kết của node cuối cùng trong DSLK là cont trỏ NULL.

Cấu trúc một node của DSLK:

```
struct node{
    int data;//phần data
    struct node *next; // link
};
```



Các thao tác trên DSLK đơn

```
struct node{
    int data;
    struct node* next;
};

node* makeNode(int x){
    node *newNode = (node*)malloc(sizeof(node));
    newNode->data = x;
    newNode->next = NULL;
    return newNode;
}
```

Các thao tác trên DSLK đơn

Duyệt DSLK

```
void duyet(node *head){
    while(head != NULL){
        cout << head->data << ' ';
        head = head->next;
    }
}
```

Đếm số node có trong DSLK

```
int size(node *head){
    int cnt = 0;
    while(head != NULL){
        ++cnt;
        head = head->next;
    }
    return cnt;
}
```

Các thao tác trên DSLK đơn

-Thêm 1 node vào đầu dslk:

Bước 1: Cập nhật phần next của node mới lưu nội dung mà con trỏ node head của DSLK đang lưu.

Bước 2: Cập nhật nội dung node head.

```
void Front(node **head, int x){
    node* newNode = makeNode(x);
    newNode->next = (*head);
    *head = newNode;
}
```

Các thao tác trên DSLK đơn

-Thêm 1 node vào cuối dslk:

Bước 1: Duyệt từ đầu danh sách tới node cuối cùng thì dừng lại.

Bước 2: Cho phần next của node cuối trỏ vào node mới được thêm vào.

```
void Back(node **head, int x){
    node* newNode = makeNode(x);
    if(*head == NULL){
        *head = newNode; return;
    }
    node* tmp = *head;
    while(tmp->next != NULL){
        tmp = tmp->next;
    }
    tmp->next = newNode;
}
```

Các thao tác trên DSLK đơn

-Thêm node vào dslk:

Bước 1: Duyệt tới node trước vị trí cần chèn gọi là node K - 1.

Bước 2: Cho phần next của node mới lưu node next của node K - 1.

Bước 3: Cho phần next của node K - 1 lưu node mới.

```
void insert(node **head, int k, int x){
    int n = size(*head);
    if(k < 1 || k > n){
        return;
    }
    if(k == 1){
        Front(head, x); return;
    }
    node *temp = *head;
    for(int i = 1; i < k - 1; i++){
        temp = temp->next;
    }
    node *newNode = makeNode(x);
    newNode->next = temp->next;
    temp->next = newNode;
}
```

Các thao tác trên DSLK đơn

-Xóa node khỏi đầu dslk:
Bước 1: Cho node head lưu node thứ 2 hiện tại trong DSLK.

Bước 2: Giải phóng node đầu tiên trong DSLK.

```
void popFront(node **head){
    if(*head == NULL) return;
    node* tmp = *head;
    *head = tmp->next;
    delete tmp;
}
```

Các thao tác trên DSLK đơn

-Xóa node khỏi cuối dslk:
Bước 1: Tìm tới node thứ 2 từ cuối về gọi là tmp.

Bước 2: Cho phần next của node tmp trỏ vào NULL.

Bước 3: Giải phóng node cuối cùng trong DSLK.

```
void popBack(node **head){
    if(*head == NULL) return;
    node *tmp = *head;
    if(tmp->next == NULL){
        *head = NULL; delete tmp;
        return;
    }
    while(tmp->next->next != NULL){
        tmp = tmp->next;
    }
    node *last = tmp->next;
    tmp->next = NULL;
    delete last;
}
```

-Xóa node ở vị trí K trong dslk

```
void erase(node **head, int k){
    int n = size(*head);
    if(k < 1 || k > n) return;
    if(k == 1) popFront(head);
    else{
        node *truoc = *head;
        node *sau = *head;
        for(int i = 1; i <= k - 1; i++){
            sau = truoc;
            truoc = truoc->next;
        }
        sau->next = truoc->next;
        delete truoc;
    }
}
```

Các thao tác tìm kiếm:

-Tìm node thứ K trong DSLK:

```
void kthNode(node *head, int pos){
    int n = size(head);
    if(pos < 1 || pos > n){
        cout << "Vi tri khong hop le !\n";
        return;
    }
    node *temp = head;
    for(int i = 1; i <= pos - 1; i++){
        temp = temp->next;
    }
    cout << temp->data << ' ';
}
```

-Tìm node có giá trị lớn nhất trong DSLK:

```
void maxNode(node *head){
    int res = -1e9;
    while(head != NULL){
        res = max(res, head->data);
        head = head->next;
    }
    cout << res << ' ';
}
```

Sắp xếp DSLK

```
void sort(node **head){
    for(node *i = *head; i != NULL; i = i->next){
        node *min = i;
        for(node *j = i->next; j != NULL; j = j->next){
            if(j->data < min->data){
                min = j;
            }
        }
        int tmp = min->data;
        min->data = i->data;
        i->data = tmp;
    }
}
```

Thuật toán sử dụng: Selection Sort

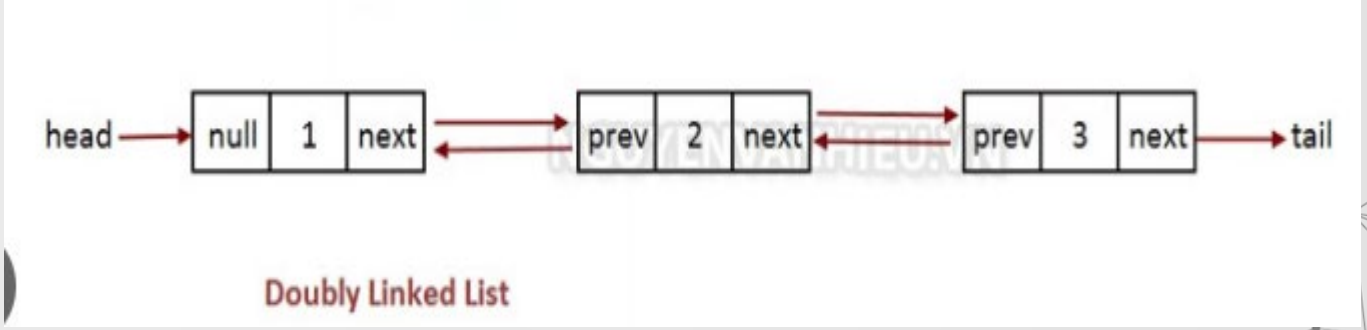
Chú ý là ta chỉ cần hoán vị phần data của 2 node còn không cần phải hoán vị next.

DSLK đôi (Double Linked List)

C++

DSLK đôi (Double Linked List)

-Ưu điểm của DSLK đôi đó là có thể di chuyển DSLK theo cả 2 chiều, tuy nhiên nó cùng cần thêm bộ nhớ để lưu con trỏ tới node liền trước cũng như như các thao tác trên DSLK đôi sẽ nhiều hơn so với DSLK đơn.



Các thao tác trên DSLK đôi

-Cấu trúc một node:

```
struct node{
    int data;
    struct node *next;
    struct node *prev;
}
```

Tạo một node mới:

```
node* makeNode(int x){
    node *newNode = (node*)malloc(sizeof(node));
    newNode->data = x;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
```

Các thao tác trên DSLK đôi

-Đếm số node có trong DSLK:

```
int size(node *head){
    int cnt = 0;
    while(head != NULL){
        ++cnt;
        head = head->next;
    }
    return cnt;
}
```

Duyệt DSLK theo chiều thuận:

```
void duyetThuan(node *head){
    while(head != NULL){
        cout << head->data << ' ';
        head = head->next;
    }
}
```

Các thao tác trên DSLK đôi:

-Thêm vào đầu DSLK:

```
void pushFront(node **head, int x){
    node* newNode = makeNode(x);
    newNode->next = (*head);
    if(*head != NULL)
        (*head)->prev = newNode;
        (*head) = newNode;
}
```

Thêm vào cuối DSLK:

```
void pushBack(node **head, int x){
    node* newNode = makeNode(x);
    if(*head == NULL){
        *head = newNode; return;
    }
    node* tmp = *head;
    while(tmp->next != NULL){
        tmp = tmp->next;
    }
    tmp->next = newNode;
    newNode->prev = tmp;
}
```

Các thao tác trên DSLK đôi:

-Thêm vào node thứ K trong DSLK:

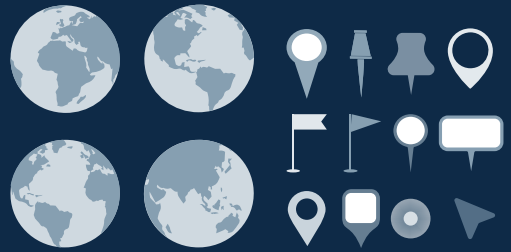
```
void insert(node **head, int k, int x){
   int n = size(*head);
   if(k < 1 || k > n){
      cout << "Vi tri chen khong hop le !\n"); return;
   }
   if(k == 1){
      pushFront(head, x); return;
   }
   node *temp = *head;
   for(int i = 1; i <= k - 1; i++){
      temp = temp->next;
   }
   node *newNode = makeNode(x);
   newNode->next = temp;
   temp->prev->next = newNode;
   newNode->prev = temp->prev;
   temp->prev = newNode;
}
```

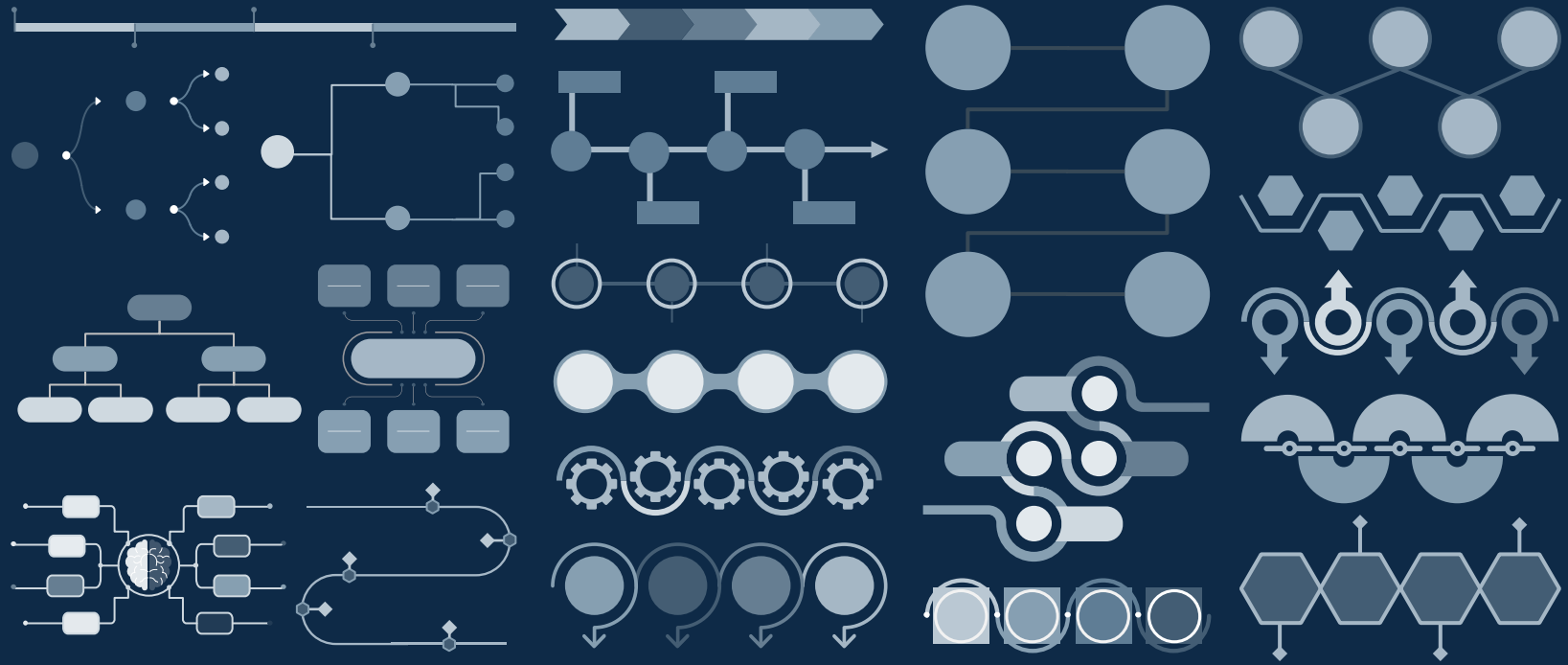
THANKS

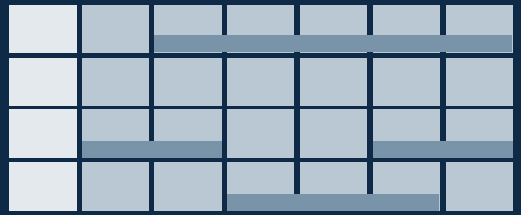
Full House Tất Cả Là Một Nhà!

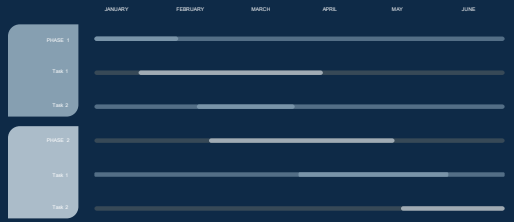


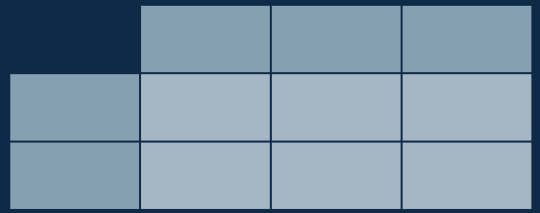


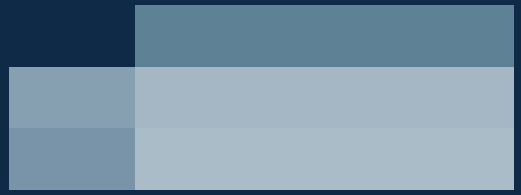



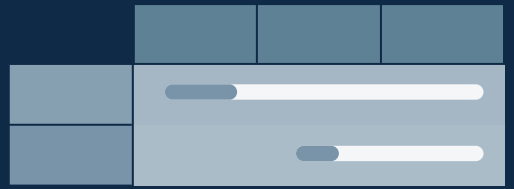


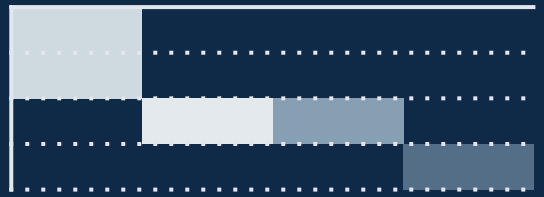


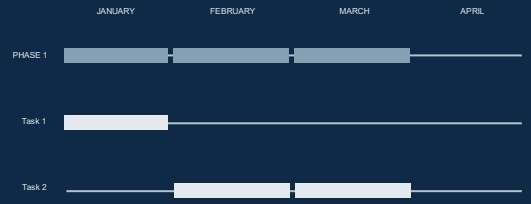

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

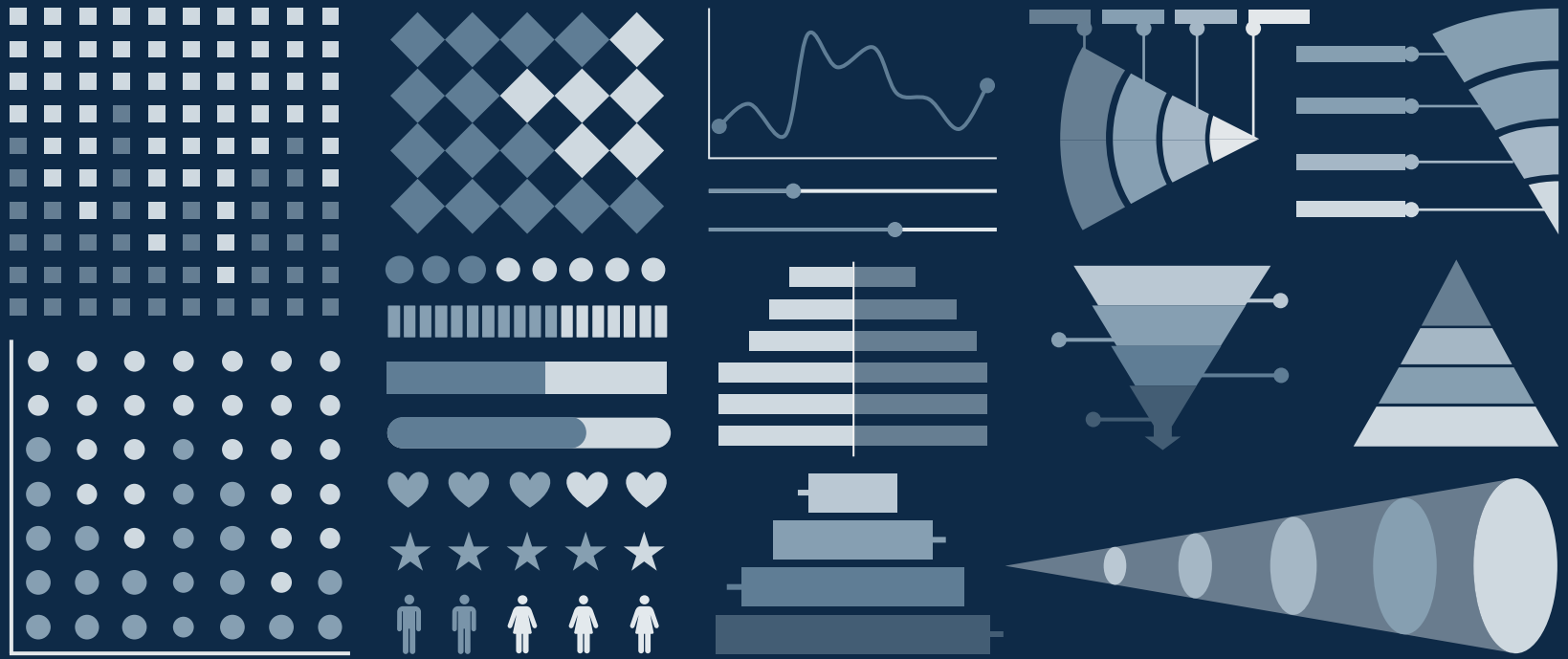









JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2



...and our sets of editable icons

You can resize these icons, keeping the quality.
You can change the stroke and fill color; just select the icon and click on the paint bucket/pen.
In Google Slides, you can also use Flaticon's extension, allowing you to customize and add even more icons.

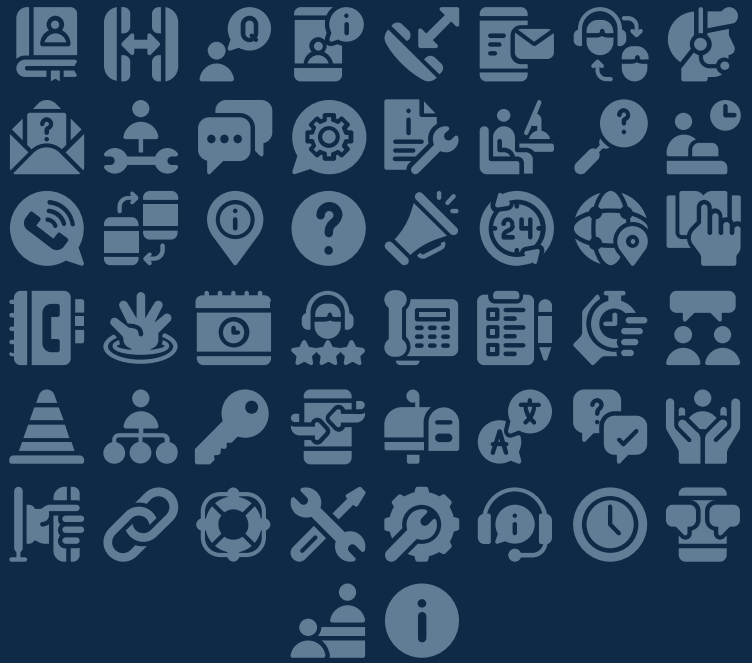
Educational Icons

Medical Icons

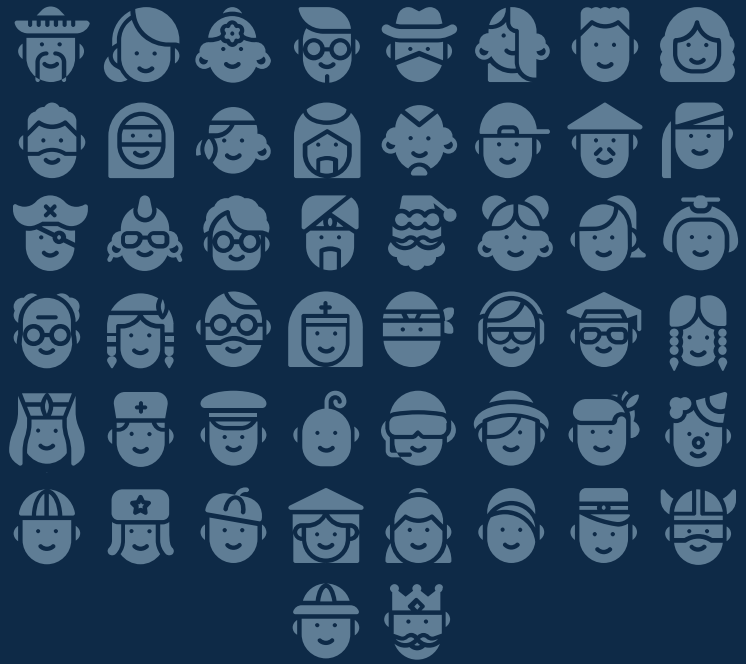
Business Icons

Teamwork Icons

Help & Support Icons



Avatar Icons



Creative process Icons

Performing Arts Icons

Nature Icons

SEO & Marketing Icons


slidesgo